2 Modelos y Arquitecturas de Agente

Autores: Juan Manuel Corchado (USAL)

2.1 Introducción

Tanto la estructura interna de un agente como la organización de un sistema multiagente están determinadas por alguna arquitectura. Este tema empieza con una revisión de las arquitecturas y modelos conceptuales más significativos en los que se puede basar el diseño de cada agente (sección 2.2). Cuando se considera una organización de agentes surge el concepto de sistema multiagente, cuya problemática desde el punto de vista de diseño del sistema compuesto de diversos agentes se presenta en la sección 2.3. En este sentido, mostramos a continuación la arquitectura estándar para el desarrollo de sistemas multiagente propuesta por FIPA (sección 2.4), y una arquitectura particular de gran difusión, denominada RETSINA (sección 2.5). Por último, la sección 2.6 presenta unas conclusiones generales sobre este capítulo.

La arquitectura determina los mecanismos que utiliza un agente para reaccionar a los estímulos, actuar, comunicarte, etc. Un hecho evidente, en este sentido, hoy en día es que existe una amplia diversidad de propuestas, casi tantas como equipos de investigación centrados en el tema (Wooldridge y Jennings 1994).

Las arquitecturas utilizadas para construir este tipo de sistemas, especifican cómo se descomponen los agentes en un conjunto de módulos que interactúan entre sí para lograr la funcionalidad requerida. Uno de los aspectos básicos que diferencia una arquitectura de otra es el método de descomposición del trabajo en tareas particulares. En este sentido hay que decir que la planificación es un área muy fuertemente ligada al mundo de la agencia. Esta área se centra en el estudio de mecanismos que permitan organizar la ejecución de acciones, y un agente no es más que un sistema que ejecuta acciones en un entorno determinado.

Los sistemas de planificación utilizan modelos de representación del conocimiento y razonamiento de tipo simbólico y su modo de actuación está definido por la necesidad de satisfacer unos objetivos básicos, para lo que elaboran un plan. Estos sistemas tienen la desventaja de que pueden requerir un elevado tiempo de respuesta. Éste es un gran inconveniente cuando se utilizan en problemas de tiempo real, ya que los algoritmos de planificación no siempre responden en un tiempo prudencial a las demandas del sistema. Esto es así debido a que los principios básicos de la planificación son indecidibles, lo que hace que no constituyan una opción del todo viable para los agentes. Estas críticas dirigidas fundamentalmente hacia el modelo simbólico utilizado, llevaron a la búsqueda de alternativas que utilizasen otro modelo de representación o razonamiento, como los reactivos o híbridos.

A continuación se presentan los agentes como sistemas intencionales así como una descripción de los tres tipos de arquitecturas que en la actualidad cubren el espectro de posibilidades que ofrece esta tecnología. Una vez analizadas las arquitecturas a nivel de agente, se detallarán aspectos relacionados con las arquitecturas en sistemas multiagente.

2.1.1 Agentes como sistemas intencionales

Los humanos a menudo utilizan afirmaciones del tipo: “Luisa lleva un paraguas porque cree que va a llover.” Esta forma de estructurar ideas permite a los psicólogos predecir el comportamiento humano y explicarlo mediante actitudes mentales, tales como creencias, deseos, temores etc. (Cañamero 1997, Grosz y Sidner 1990). El filósofo Daniel Dennett (1987) denomina “sistemas intencionales” a aquellas entidades “cuyo comportamiento se puede predecir por el método de atribuir creencias, deseos y perspicacia racional”.

¿Qué objetos se pueden describir desde esta postura intencional? Autores como Seel (1989), Rosenschein y Kaelbling (1986) han llegado a la conclusión, de que se puede definir cualquier cosa. Si el sistema es muy sencillo, atribuir descripciones intencionales consistentes con el comportamiento observado no resulta de gran utilidad. Sin embargo, en sistemas complejos las nociones intencionales son herramientas de abstracción, que proporcionan una forma adecuada y familiar de describir, explicar y predecir el comportamiento de éstos sistemas.

Por tanto un agente puede verse como un sistema capaz de definirse correctamente mediante posturas intencionales (Bratman, 1990). Un agente contiene actitudes que proporcionan información (actitudes de información) y pro-actitudes.

Actitudes de Información	**Pro-actitudes**
Creencia Conocimiento	Deseo Intención Obligación Compromiso Elección ...

Tabla 2.1: Tipos de actitudes.

Las actitudes de información están relacionadas con el conocimiento que un agente tiene sobre su entorno, mientras que las pro-actitudes son aquellas que guían de algún modo las acciones del agente. Precisamente, la elección de cual es la combinación de actitudes más adecuada para caracterizar los agentes es motivo de grandes discusiones. Parece razonable que dicha combinación posea, al menos, una actitud de información y una pro-actitud (Wooldridge *et al.*, 1995). Un agente toma decisiones y posee intenciones en base a la información disponible de su entorno, por tanto, la relación entre las dos categorías de actitudes parece directa.

Como ya se indicó, no hay un consenso claro entre el mundo de la inteligencia artificial y la comunidad de filósofos acerca de cual es la combinación entre las actitudes de información y las pro-actitudes más adecuadas para definir un agente. A continuación se incluye una revisión en la cual se indica qué actitudes y pro-actitudes incorporan diferentes autores para caracterizar un agente (Wooldridge et al., 1995).

- Conocimiento y acción: Moore (1990) es un pionero en el uso de lógicas para capturar ciertos aspectos de los agentes. Se centra en definir qué datos debería conocer un agente para así proyectar el conjunto de precondiciones en el de acciones. De esta manera un agente podría ejecutar las acciones en un orden

determinado como consecuencia de un estímulo. Moore formalizó el "modelo de habilidad" con una lógica que modela el conocimiento, y una lógica dinámica para el modelado de las acciones. Este formalismo permite que un agente que posea información incompleta sobre cómo alcanzar algún objetivo, pueda hacerlo experimentando y ejecutando acciones en un orden concreto.

- Intención: Cohen y Levesque (1990) utilizan dos actitudes básicas: creencias y objetivos. Las actitudes, como la intención, se definen en términos de las anteriores. Originalmente, su formalismo se utilizó para desarrollar la "Teoría de Intención". Se ha demostrado posteriormente que la lógica propuesta por estos autores es adecuada para generar mecanismos de razonamiento que los agentes pueden utilizar para el análisis de conflictos, la cooperación y, también, como fundamento teórico de cooperación en la resolución de problemas (Cohen y Levesque, 1990 y 1995).
- Creencias, Deseos, Intenciones: Rao y Georgeff (1991) han desarrollado una estructura lógica para la teoría de agentes basada en tres instancias primitivas: creencias, deseos e intenciones. La sección 2.2.1.1 profundiza en este modelo.
- Intenciones, Creencias, Conocimiento, Habilidades: Singh (1994) ha desarrollado una familia interesante de lógicas para representar intenciones, creencias, conocimiento, habilidades y comunicación utilizando una extensión del modelo de tiempo, caracterizado por incorporar un formalismo extremadamente rico y complejo.
- Wooldridge (1992) ha desarrollado una familia de lógicas para representar las propiedades de los sistemas multiagentes. Sus formalismos se usan en la especificación y verificación de estos sistemas. Desarrolló un modelo simple y genérico de sistemas multiagentes realistas, y mostró ejemplos de cómo estas lógicas se utilizan en la especificación y verificación de protocolos de cooperación.

Estas son algunas de las propuestas más relevantes y quizás la base de la mayor parte de arquitecturas de agente deliberativas e híbridas desarrolladas.

2.2 Arquitecturas para construir agentes

En esta sección se presentan las arquitecturas deliberativas con más detalle, así como las reactivas y las híbridas, generadas por la combinación de aspectos de las dos anteriores.

2.2.1 Deliberativas

Son aquellas arquitecturas que utilizan modelos de representación simbólica del conocimiento. Suelen estar basadas en la teoría clásica de planificación. Estos agentes parten de un estado inicial y son capaces de generar planes para alcanzar sus objetivos (Maes, 1989). En estos sistemas parece aceptada la idea de que es necesario dotar a los agentes de un sistema de planificación que se encargue de determinar qué pasos se deben llevar a cabo para alcanzar sus objetivos. Por tanto, un agente deliberativo (o con una arquitectura deliberativa) es aquel que contiene un modelo simbólico del mundo, explícitamente representado, en donde las decisiones se toman utilizando mecanismos de razonamiento lógico basados en la correspondencia de patrones y la manipulación simbólica, con el propósito de alcanzar los objetivos del agente

Cuando se decide implementar una arquitectura deliberativa hay que buscar, en primer lugar, una descripción simbólica adecuada del problema, e integrarla en el agente, para que este pueda razonar y llevar a cabo las tareas encomendadas en el tiempo

preestablecido. Aunque parece una cuestión trivial, debido a la complejidad de los algoritmos de manipulación simbólica, es un aspecto al que hay que prestar mucha atención, especialmente si se tiene en cuenta que los agentes se desenvuelven en dominios reales, en los que frecuentemente tienen que responder a los estímulos en tiempo real.

Los agentes intencionales se pueden implementar utilizando arquitecturas deliberativas. Estos agentes están dotados de modelos de planificación capaces de generar planes a partir de las creencias e intenciones (Jennings, 1993). El siguiente apartado está dedicado a explicar con detalle la ya citada arquitectura deliberativa BDI (*Belief, Desire, Intention*), que es la más estudiada y posiblemente más extendida (Rao *et al.*, 1995).

2.2.1.1 Arquitectura BDI

La arquitectura BDI (Belief, Desire, Intention) está caracterizada por el hecho de que los agentes que la implementan están dotados de los estados mentales de *Creencias*, *Deseos* e *Intenciones* (Haddadi y Sundermeyer, 1996). El éxito de esta arquitectura se debe posiblemente a que combina elementos interesantes: un modelo filosófico del razonamiento humano fácil de comprender, un número considerable de implementaciones (Georgeff *et al.*, 1987) y una semántica lógica abstracta y elegante, que ha sido aceptada por gran parte de la comunidad científica (Rao y Georgeff, 1998; Georgeff *et al.*, 1999; Schild, 1999).

Las nociones de complejidad y cambio tienen un gran impacto en la forma en que se construyen los sistemas computacionales y por tanto en los agentes. Los agentes y en particular los agentes BDI incorporan componentes que permiten el desarrollo de sistemas que se integren adecuadamente en el mundo real (Georgeff, 1998). Muchas de las aplicaciones de sistemas informáticos son algorítmicas y trabajan con información exacta. Pero la mayoría de las aplicaciones requieren sistemas más complejos, sistemas capaces de relacionarse con un entorno cambiante y con un cierto grado de incertidumbre. Los agentes y los sistemas multiagentes tienen por tanto que ser capaces de proporcionar soluciones a este tipo de problemas. El modelo BDI se ha desarrollado para proporcionar soluciones en entornos dinámicos o inciertos, en los que el agente o los agentes sólo tienen una visión parcial del problema (el acceso a la información está limitado) y posiblemente manejen un número limitado de recursos (recursos informáticos finitos). Las creencias, los deseos, las intenciones, y los planes son una parte fundamental del estado de ese tipo de sistemas.

En el campo de la inteligencia artificial, las creencias representan el conocimiento que se tiene del entorno. Desde un punto de vista informático, son la forma de representar el estado del entorno, por ejemplo el valor de una variable, los valores de una base de datos relacional, o expresiones simbólicas del cálculo de predicados. Las creencias son esenciales en esta arquitectura. En medios dinámicos es necesario mantener información sobre los eventos pasados al mismo tiempo que se debe permitir su adaptación y evolución, por lo que una buena gestión de estos elementos es vital para construir agentes eficientes.

Los deseos (objetivos) son otro componente esencial en esta arquitectura. En términos informáticos, un objetivo puede simplemente ser el valor de una variable, un registro, o una expresión simbólica en alguna lógica. Un objetivo representa un estado final deseado. El software convencional está “orientado a la tarea” en lugar de “al objetivo”,

de forma que cada tarea (o subrutina) se ejecuta sin ningún recuerdo de por qué ha comenzado su ejecución. Esto significa que el sistema no puede recuperarse ante fallos automáticamente (a menos que esto sea específicamente codificado por el programador) y no puede descubrir ni aprovechar oportunidades que surjan inesperadamente. Por ejemplo, la razón de que un humano se recupere de, resuelva la pérdida de un tren o un pinchazo inesperado de la rueda de su coche, es porque conoce dónde está (a través de sus creencias) y recuerda qué quiere conseguir (a través de sus objetivos). La semántica fundamental de los objetivos, sin tener en cuenta cómo se representa computacionalmente, se reflejaría en una lógica de los deseos.

Para alcanzar los objetivos propuestos, a partir de las creencias existentes es necesario definir un mecanismo de planificación que nos permita identificar las intenciones. En este sentido, tenemos que tener claro que los agentes están inmersos en sistemas dinámicos, en los que en ocasiones será necesario decidir si replanificar o no, ante cambios en el entorno, durante la ejecución del plan inicialmente seleccionado. Los adherentes a la teoría de la decisión clásica dicen que siempre se debería replanificar, mientras que los diseñadores de software convencional orientado a la tarea obligarían a continuar hasta el final con el plan inicialmente previsto. ¿Cuál es el enfoque correcto? Posiblemente ni la teoría de la decisión clásica ni la aproximación convencional son, por sí solas, apropiadas. El sistema necesita comprometerse con los planes y subobjetivos, pero también ser capaz de reconsiderar éstos en los momentos clave. Estos planes vinculados a la consecución de un objetivo constituyen las intenciones del agente. Las intenciones son simplemente un conjunto de caminos de ejecución ("hebras", en inglés *threads*) que pueden ser interrumpidos de una forma apropiada al recibir información acerca de cambios en el entorno (Kinny y Georgeff, 1993; Georgeff, 1998).

A continuación se presentan una serie de aspectos formales relacionados con la arquitectura BDI, una arquitectura abstracta para la implementación de agentes y algunos sistemas prácticos que aplican esta arquitectura. La aplicación de esta arquitectura para la construcción de sistemas multiagentes se presenta en el capítulo de metodologías.

2.2.1.1.1 Formalismo teórico del modelo BDI

Las actitudes se representan en un formalismo basado en la lógica modal y el concepto de "mundos posibles" (*possible worlds*). De forma intuitiva, la lógica modal se puede definir como aquella que permite razonar sobre lo que podría ser, o se cree que es, en lugar de lo que es realmente. Es decir, permite representar conocimiento del tipo "Es posible que Luis tenga apendicitis". Esto es, desde la situación actual, se puede llegar a una nueva en la que Luis tenga esta enfermedad, aunque también sería posible que se llegase a otras situaciones (mundos) en las que Luis no tuviese apendicitis.

Para conseguir que la lógica incluya estos significados se maneja una semántica de mundos posibles con una relación de accesibilidad entre ellos. Dicha semántica hace necesario ampliar la sintaxis con la incorporación de nuevos operadores unitarios: el operador de necesidad y el operador de posibilidad (Chellas, 1980; Hughes *et al.*, 1996). La relación de accesibilidad enlaza la situación actual (el mundo actual) con todas las que son posibles a partir de ella. De este modo los conceptos de verdad necesaria y verdad posible se definen a partir de esta relación de accesibilidad. Una proposición es necesariamente verdadera en un mundo si es verdadera en todos los mundos que son accesibles desde él (lo que incluye el caso de que no existiese ningún mundo accesible).

Una proposición es posiblemente verdadera en un mundo si es verdadera en alguno de los mundos accesibles desde él (Chellas, 1980).

Los agentes BDI se pueden modelar utilizando una estructura, basada en la lógica de mundos posibles, denominada árbol temporal con múltiples futuros y un solo pasado (Rao *et al*., 1991), tal y como se muestra en la Figura 2.1. Cada nodo del árbol es una situación (o mundo posible) y las ramas del árbol pueden verse como las opciones disponibles para el agente en cada momento del tiempo. Para esta estructura de mundos posibles se definen varias relaciones de accesibilidad (distintas modalidades). Esto es, para cada situación, se definen una serie de mundos accesibles (nuevas situaciones a las que el agente puede llegar) desde el punto de vista de las creencias (los mundos que el agente considera posibles), de los deseos (mundos que el agente desea alcanzar) y de las intenciones (aquellos mundos que el agente ha decidido que intentará alcanzar) (Rao y Georgeff, 1995). Como ya se indicó, los deseos podrían ser incompatibles, pero este modelo exige que al menos sean consistentes y, además, como se comentará posteriormente, el agente debe creer que sean alcanzables.

Semánticamente, podría decirse que el agente, situado en un mundo accesible a partir de sus creencias, cambia a un mundo accesible por objetivos al desear nuevos caminos de acción. De ese estado evoluciona a otro accesible por medio de las intenciones al comprometerse a realizar las acciones deseadas (Rao *et al*., 1991).

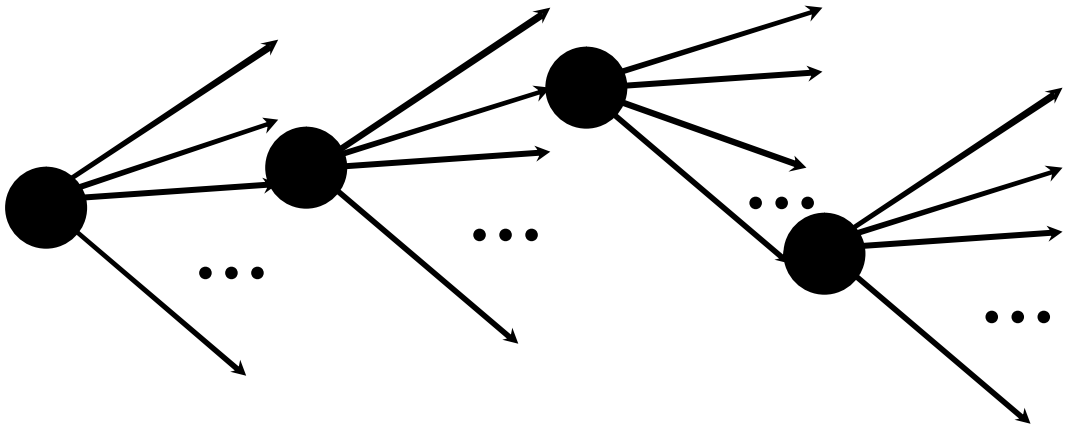

Figura 2.1: Árbol de mundos posibles.

Basándose en el formalismo teórico anteriormente descrito, se definen las relaciones que deben existir entre las creencias, los deseos y las intenciones del agente. Algunas de las más importantes se enumeran a continuación de manera intuitiva:

- Compatibilidad entre creencias y objetivos. Si el agente adopta el deseo de alcanzar un objetivo, debe creer que en alguno de los mundos accesibles por la relación de creencia, dicho objetivo es cierto.
- Compatibilidad entre objetivos e intenciones. Previamente a que el agente adopte una fórmula como intención debe haberla adoptado como deseo.
- Las intenciones conducen a acciones. Si una de las intenciones es una acción primitiva o simple, el agente la ejecuta. Es decir, no se pospone la ejecución de una acción simple.

- Relación entre creencias e intenciones. El agente conoce (cree en) sus propias intenciones.
- Relación entre creencias y objetivos. El agente conoce sus objetivos o deseos.
- No hay retrasos infinitos. Cuando un agente adopta una intención, sigue con ella hasta algún momento del futuro, es decir, no puede haber una parada infinita en el proceso de alcance de un determinado objetivo (esto es, debe haber progreso).

Un aspecto importante de la arquitectura BDI es la noción de compromiso con las decisiones previas. El compromiso proporciona cierta estabilidad al proceso de razonamiento. De este modo, se economiza el esfuerzo informático y, por lo tanto, mejora la ejecución global. Un compromiso tiene dos partes: una es la condición que el agente está comprometido a mantener, llamada condición comprometida; y la segunda es la condición bajo la cual el agente renuncia al compromiso, llamada condición de terminación (Rao y Georgeff, 1995).

Como el agente no tiene control directo sobre sus creencias y deseos, no hay forma de que pueda adoptar o realizar un compromiso estratégico sobre estas actitudes. Sin embargo, un agente puede elegir qué hacer con sus intenciones. Así, se restringe la condición de compromiso sobre las intenciones. Un agente puede comprometerse a una intención basándose en que el objetivo de la intención es satisfecho en un camino futuro o todos los caminos futuros, provocando diferentes condiciones de compromiso y por lo tanto, diferentes comportamientos dinámicos. Las diferentes condiciones de terminación originan nuevos comportamientos.

De este modo, las intenciones actuales del agente guían o influyen en sus decisiones sobre futuras intenciones. Dependiendo de cómo afecten las intenciones pasadas a las futuras, se identifican varios tipos de agentes:

- Ciego. El agente mantiene sus intenciones hasta que sabe que las ha alcanzado. Por lo tanto, si es necesario rechazará las creencias o deseos que contradigan sus compromisos.
- Firme. El agente mantiene sus intenciones mientras crea que tiene opciones de alcanzarlas.
- Imparcial. El agente mantiene sus intenciones mientras éstas se corresponden con sus deseos, es decir, mientras el deseo o deseos que dieron lugar a esa intención no cambian.

Es importante tener en cuenta que todos los aspectos definidos aquí de un modo intuitivo han sido formalizados empleando elementos de la lógica modal. Por lo tanto, existe una base formal en la que definir y demostrar las propiedades que han sido citadas (Rao *et al.*, 1991). El propósito de esta formalización es construir sistemas prácticos y verificables. Si para un dominio de aplicación se conocen los cambios del entorno y los comportamientos esperados del sistema, se puede usar esta formalización para especificar, diseñar y verificar agentes, que cuando estén en su entorno exhibirán los comportamientos deseados.

El formalismo descrito en los párrafos anteriores se puede ampliar para permitir la inclusión de librerías de planes, que definirán métodos estándar para la consecución de determinados objetivos (Cavendon *et al.*, 1995). Estos planes se componen de una condición de invocación (el objetivo que permiten conseguir) y un cuerpo (una secuencia de acciones para alcanzar el objetivo o incluso subobjetivos, que pueden tener o no asociados nuevos planes).

De este modo y de forma similar al modelo de creencias, deseos e intenciones, se define un mundo de planes, aunque no es necesaria una relación de accesibilidad entre mundos puesto que los planes son estáticos, es decir, los mismos planes están disponibles en cada una de las situaciones posibles.

Evidentemente, debe existir una interrelación entre las intenciones de un agente y los planes de que dispone. Estas relaciones se han definido formalmente (Cavendon *et al.*, 1997), siendo las más importantes las citadas a continuación:

- Las intenciones de un agente están restringidas por sus planes, es decir, las acciones se deben alcanzar exclusivamente utilizando los planes de la librería de planes del agente.
- Si un agente tiene la intención de alcanzar el objetivo ϕ, debe adoptar también la intención de ejecutar el cuerpo del plan que permite alcanzar ϕ.
- Por otro lado, se espera que si un agente tiene un plan para alcanzar el objetivo ϕ, dentro de sus creencias esté el hecho de que el plan logrará el objetivo fijado (ϕ).

2.2.1.1.2 Arquitectura Abstracta

La arquitectura propuesta en Rao y Georgeff (1995) se basa en la construcción de sistemas de agentes cuyas estructuras de datos se corresponden con cada uno de los componentes de los agentes BDI: creencias, deseos e intenciones, como se muestra en la Figura 2.2. Cada uno de los componentes descritos se guarda aisladamente, o sea, existe una lista de creencias, otra de deseos y otra de intenciones. Además, se trabaja con una secuencia de eventos o acontecimientos ocurridos en el entorno. Cada uno de ellos se implementa como una cola, esto es, mediante una estructura de datos lineal con comportamiento FIFO (primero en llegar primero en salir).

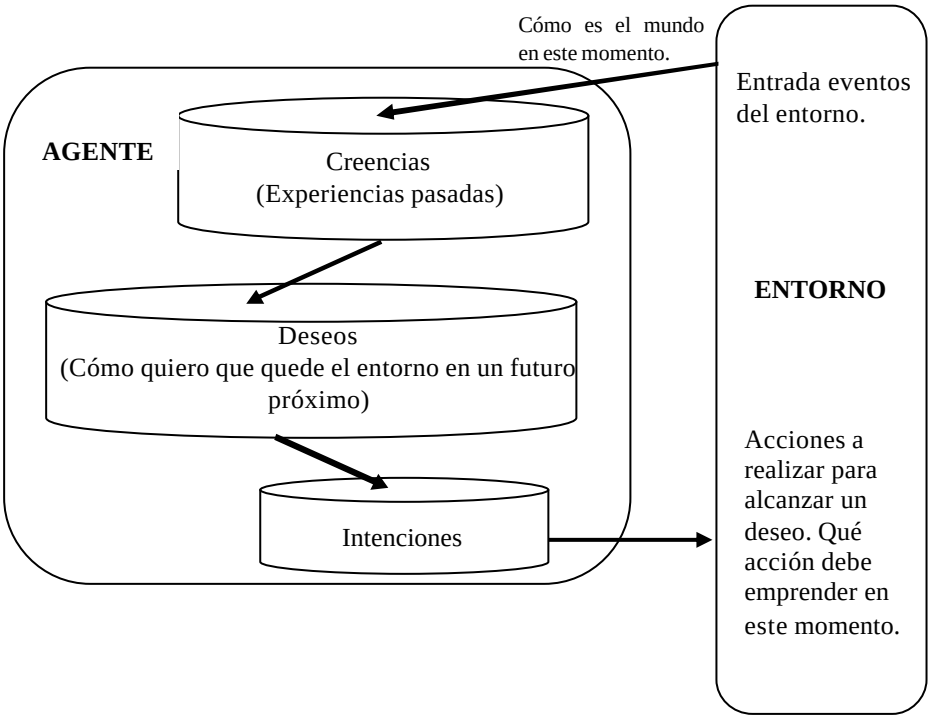

Figura 2.2. Arquitectura BDI.

Por lo que respecta a la utilización de estos datos para lograr la reactividad en el agente, parece evidente que la implementación no puede estar basada en métodos tradicionales como la demostración de teoremas. La razón es que, aunque el método pudiese ser ampliado para dar cobertura a todos los aspectos lógico-formales antes descritos, no podría determinarse el límite máximo para el tiempo de computación. Esta característica no permitiría garantizar la reacción en tiempo real ante un entorno cambiante y, por tanto, comprometería la supervivencia del agente.

Por ello, la arquitectura propuesta realiza iterativamente una serie de pasos, cada uno de ellos con una duración limitada y por tanto, con posibilidad de reacción ante el entorno en un tiempo que, a nivel práctico, es suficiente. Al comienzo del ciclo, se lee la cola de eventos y se devuelve una lista de opciones. Se seleccionan aquellas que se deben adoptar y se añaden a la cola de intenciones. A continuación, se ejecutan todas las intenciones que impliquen la realización de una acción simple. En este momento se comprueba si existen nuevos eventos en el entorno y se incorporan a la cola de eventos. Por último, el agente modifica las estructuras de deseo e intención, eliminando los ya satisfechos y los que son imposibles de alcanzar. Un esquema de este ciclo de ejecución, tomado de Rao y Georgeff (1995), sería:

```
Interprete-BDI
Inicializar_estado();
Repetir
      Opciones:= generador_opciones (cola_eventos);
      Opciones_seleccionadas :=  decidir (opciones);
      Actualizar_intenciones (opciones_seleccionadas);
      Ejecutar();
      Leer_nuevos_eventos_externos();
      Eliminar_intenciones_alcanzadas();
      Eliminar_intenciones_imposibles();
Fin repetir
```

La arquitectura presentada por Rao y Georgeff (1995) es una idealización que representa los conceptos teóricos, aunque no parece completamente implementable a niveles prácticos. Por ejemplo, no se hace referencia a cómo desarrollar los procesos de generación o de selección de opciones de forma suficientemente eficiente como para lograr las respuestas en un rango de tiempo adecuado.

Por todo ello, dentro de este enfoque general, a la hora de la implementación se han introducido algunas limitaciones para mejorar el modo de razonamiento que restringen el poder expresivo de la arquitectura pero mejoran su rendimiento a niveles prácticos. Por ejemplo, al definir las creencias hay que evitar el uso de disyunciones o implicaciones. Además, los agentes suelen incorporar una librería de planes ya definidos (Cavendon *et al.*, 1995).

2.2.1.1.3 Algunos ejemplos

Utilizando el formalismo teórico y la arquitectura descritos previamente se han implementado distintas aplicaciones. Entre ellas cabe destacar un sistema gestor de

tráfico aéreo, un sistema para la gestión de procesos de negocios y un sistema de modelado de combate aéreo.

El sistema gestor de tráfico aéreo (OASIS) [Georgeff *et al*., 1998] consta de un gran número de agentes construidos siguiendo los esquemas marcados anteriormente y, entre otras cosas, permite seleccionar una trayectoria de vuelo de entre varias posibles. Los agentes ayudan a seleccionar el mejor camino para minimizar el consumo de carburante, aumentar la productividad del vuelo, etc. y, a la vez, garantizar que se cumple para todos ellos el tiempo previsto de llegada. El sistema se ha utilizado en el aeropuerto de Sidney.

La arquitectura OASIS divide las tareas de control del tráfico aéreo en grandes bloques y asigna un agente para resolver cada uno de esos subproblemas. Cada uno de los agentes resuelve su parte de modo independiente y coopera con el resto para definir el comportamiento global del sistema. En una ejecución habrá un agente por cada uno de los aviones implicados, además de otros agentes que actúan como secuenciadores, modeladores de viento, coordinadores y controladores de trayectoria.

SWARMM es un sistema de modelado de combate aéreo que permite codificar las tácticas adoptadas por los pilotos en escenarios de combate y crear agentes que pueden participar con pilotos reales en estudios de simulación. El sistema modela tanto los aspectos físicos del avión (armas, sensores, rendimiento…) como el conocimiento táctico y la forma de razonar de los pilotos (Rao *et al*., 1992).

Corchado *et al.* (2004) han desarrollado una arquitectura deliberativa sobre el marco metodológico que proporcionan los sistemas de razonamiento basados en casos. El sistema de razonamiento guía la construcción de planes a partir de las creencias y las intenciones. En función del problema a resolver y de las técnicas utilizadas en cada una de las cuatro etapas definidas por el sistema de razonamiento, el agente puede incluso planificar en tiempo de ejecución utilizando cálculo variacional [Bédia *et al.*, 2002]. Este modelo se ha utilizado en el desarrollo de aplicaciones industriales para en el sector de la construcción [Corchado y Laza, 2003] y en el desarrollo de aplicaciones para dispositivos móviles [Corchado *et al.*, 2004].

Otro ejemplo que podemos presentar es la arquitectura AFRAS (Carbó *et al.* 2003) que mediante la inclusión de lógica borrosa en el razonamiento interno del agente gestiona el problema de la reputación en entornos de mercados electrónicos. Los agentes que componen el sistema denominado *AFRAS* (*A Fuzzy Reputation Agent System*) están dedicados a la compra/venta de bienes y servicios. A los tradicionales papeles de comprador y vendedor, la selección del comercio mediante la cooperación con otros agentes introduce un nuevo tipo de papel: el intermediario que genera recomendaciones (recomendante). El sistema de agentes *AFRAS* pretende emular el funcionamiento de la sociedad humana tanto en la gestión y comunicación de esta reputación genérica subjetiva como en el regateo informal que tiene lugar entre particulares. En consecuencia, con este objetivo de reflejar esta relación social con un sistema de agentes, un único agente *AFRAS* por sí mismo trata de reflejar el razonamiento y el comportamiento propio de una persona. Así las decisiones necesarias para seleccionar un comercio se toman de acuerdo a atributos que caracterizan el comportamiento de un agente como el de un ser humano. Así el *egoísmo*, la *susceptibilidad* y la *timidez* de un agente determinan su comportamiento.

La arquitectura que adoptan los agentes de AFRAS está inspirada en el paradigma deliberativo. La arquitectura de AFRAS también tiene tres distintos niveles de abstracción. Estos se corresponden con el modelo del mundo, social, y mental que

contiene cada agente. Estos modelos definen una arquitectura por capas, que se activa de forma ascendente y se ejecuta de forma descendente. Cada capa utiliza los predicados del nivel inmediatamente inferior, cuanto más alta está en la jerarquía, más complejas y abstractas son sus competencias. En la Figura 2.3 se muestran estos niveles y su relación con los atributos que definen al agente al realizar éste una compra.

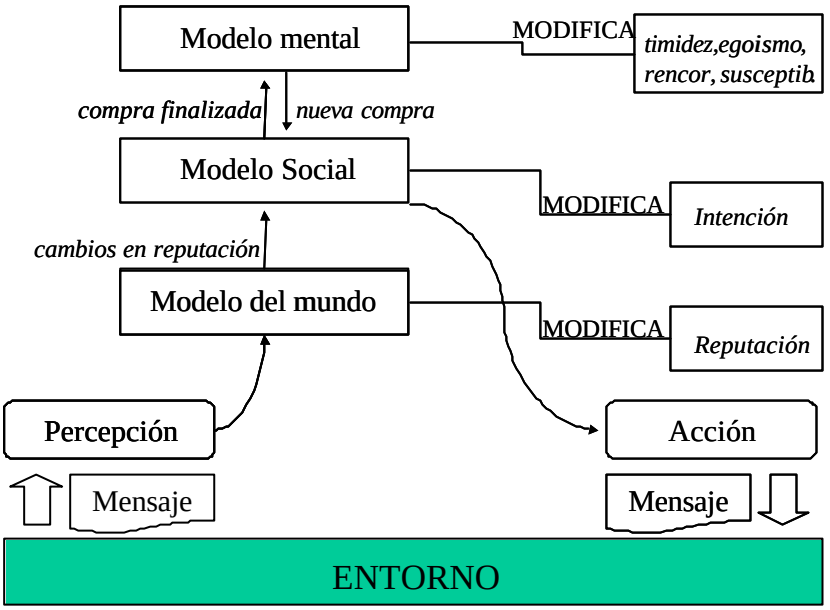

Figura 2.3. Funcionamiento del modelo conceptual de los agentes AFRAS

El nivel inferior, el modelo del mundo, se ocupa de los mensajes que recibe el agente (percepciones) actualizando la visión particular de ese agente sobre el resto de agentes en el sistema representada en forma de creencia. Así las recomendaciones recibidas, la reputación de los demás agentes, las ofertas recibidas y las contraofertas enviadas constituyen las creencias que maneja un agente comprador de AFRAS para seleccionar un comercio y negociar con él. Este módulo también toma decisiones binarias sobre cuestiones que le plantea el nivel inmediatamente superior sobre la confianza que se infiere de la reputación. Las decisiones que caen bajo la responsabilidad de este modelo consisten en preguntar o no por referencias sobre terceros a un agente dado, responder una pregunta de este tipo, iniciar una negociación de compra con un cierto vendedor, aceptar o rechazar una oferta y abandonar la negociación. Las decisiones sobre las ofertas en la negociación se toman de acuerdo a los deseos del agente, expresados en la función de preferencias interna del agente. Por su parte, las decisiones relativas a la selección del comercio se toman de acuerdo al nivel de exigencia de reputación que el agente requiere para preguntar, responder y comprar. Estos niveles representan el egoísmo, la timidez y la susceptibilidad de los agentes.

El nivel intermedio de abstracción, el modelo social, se ocupa de la secuencia de interacciones implicadas en una compra. Controla los diferentes cursos de acción que está siguiendo un agente. Estas constituyen las intenciones con las que está comprometido, ya que las intenciones son las acciones básicas que puede tomar el agente, las cuales son siempre interacciones sociales con los otros agentes del sistema.

Preguntará al modelo del mundo si debe preguntar/responder/comprar a un determinado agente.

El modelo mental del agente se ocupa de la adaptación de la forma de ser de los agentes mediante la modificación de los valores de los atributos que caracterizan el comportamiento del agente a semejanza humana, como pueden ser: el egoísmo, la timidez, la susceptibilidad, el rencor, etc. Su valor afectará a la forma en que los otros dos modelos tomarán las decisiones. Estos atributos mentales cambiarán en función de cómo evolucionen los resultados obtenidos de la ejecución de sucesivas compras. Por esta razón, el modelo social activará a este otro modelo cuando los resultados de la evaluación de una compra estén disponibles.

2.2.2 Reactivas

Los numerosos problemas que lleva asociados utilizar una representación simbólica del conocimiento han conducido al estudio de modelos más efectivos de representación del conocimiento (Bonasso *et al.*, 1995). Las arquitecturas reactivas se caracterizan por no tener como elemento central de razonamiento un modelo simbólico y por no utilizar razonamiento simbólico complejo (Brooks, 1990; Keith *et al*., 1997).

Un ejemplo típico de estas arquitecturas es la propuesta de Roodney Brooks, conocida como *arquitectura de subsunción* (Brooks, 1991). Esta arquitectura se basa en las hipótesis de que la inteligencia es una propiedad emergente de ciertos sistemas complejos y de que ello permite generar comportamientos inteligentes sin necesidad de construir un modelo simbólico. Las arquitecturas de subsunción manejan jerarquías de tareas que definen un comportamiento. Suelen estar organizadas en jerarquías de capas, de menor a mayor nivel de abstracción (Brooks, 1991).

La mayor aplicación de este tipo de arquitecturas se ha centrado en el desarrollo de controladores en robótica. Los robots se pueden considerar como agentes reales (no software) que actúan en un entorno cambiante. Precisamente la necesidad de actuar en un entorno impredecible y altamente cambiante dificulta la adopción de una arquitectura deliberativa ya que las necesidades de replanificación y de continua adaptación del plan a la realidad hacen muy difícil que una arquitectura de este tipo responda con suficiente agilidad.

2.2.2.1 Agentes reactivos en robótica

La robótica es quizás el campo de estudio en el que el concepto de agente es más directamente visualizable: un agente es el equivalente a un robot. Un robot es un agente (en el sentido más estricto) que además existe en la realidad. Es decir, es un agente real o físico, a diferencia de un agente que únicamente existe en una computadora (agente software).

La arquitectura de un agente en robótica viene definida por la arquitectura de control del robot (Molina y Matellán, 1996). Entre los diferentes tipos de arquitecturas de control propuestas a lo largo de los últimos años existen dos perspectivas mayoritarias y todo un conjunto de arquitecturas híbridas. Evidentemente, el paralelismo con las arquitecturas de agentes es obvio: la perspectiva reactiva es aquella que mantiene la posibilidad de alcanzar la consecución de objetivos complejos basándose únicamente en un conjunto de acciones simples, denomínense instintos, módulos, controladores o comportamientos, y la perspectiva más deliberativa es aquella que necesita tener un modelo interno del mundo sobre el que razonar. La segunda de estas arquitecturas obliga a una representación fehaciente del mundo. Únicamente, a partir de resultados

intermedios de razonamiento sobre dicha representación será posible la resolución de problemas complejos.

En ambos casos es necesario identificar un conjunto de acciones (las habilidades de un agente) que permitan al robot (agente real) interaccionar con el entorno en el que se encuentra inmerso. Las diferentes arquitecturas aparecen a la hora de integrar todas las acciones para dar respuesta a problemas complejos. Habitualmente el conjunto de habilidades necesita de la existencia de distintos niveles jerárquicos (agrupación de las habilidades para conseguir un objetivo concreto, el equivalente a una intención) en la estructura de control.

La definición de los distintos niveles de control se basa en las características de las tareas que se desean realizar sobre la plataforma de la que se dispone (Braitenberg 1984). Una tarea puede, en términos coloquiales, definirse como sencilla o compleja, en función de la capacidad del robot para realizarla. Por ejemplo, al pedir a un robot ejecutar un giro de 90º sobre sí mismo, si es cilíndrico con dos ruedas motrices la tarea resulta muy sencilla; pero, si consideramos un robot con patas, ese mismo giro necesita de una composición de movimientos compleja que no es realizable mediante sencillos algoritmos de control.

Siguiendo este razonamiento se pueden distinguir dos niveles de control (Figura 2.4). El nivel más alto realiza el razonamiento a largo plazo, describiendo un plan para alcanzar la tarea compleja que se desea realizar a base de ir dividiéndola en sub-acciones más sencillas, y se encarga de que cada una de estas sub-acciones alcance el sub-objetivo que se ha dispuesto, es decir, realiza la monitorización de la ejecución del plan. El nivel más bajo se encargará de la realización de las tareas más sencillas, resolviendo los problemas más inmediatos con los que se puede encontrar el robot y que no requieren de una planificación.

Figura 2.4. Niveles de control.

Las distintas arquitecturas de control desarrolladas difieren en el peso específico dado a cada uno de los niveles; en los casos más extremos, constan de un único nivel. La arquitectura deliberativa únicamente considera la existencia de un "nivel alto". En el nivel alto, el agente razonador necesita una representación detallada del entorno en el que se va a mover. A partir de dicha representación desarrolla un plan para salvar los problemas que plantea el entorno. Si sólo se considera la existencia un nivel alto, es necesario recurrir a la replanificación cada vez que el entorno se ha modificado y difiere de la representación interna del robot.

La arquitectura reactiva únicamente considera la necesidad de un conjunto de acciones muy básicas (Brooks, 1986, 1991). Desde esta perspectiva, el robot es más robusto,

pues sólo necesita representar el conjunto de acciones básicas -a veces denominadas instintos o comportamientos- que componen el nivel bajo y únicamente hay que decidir cuál de esas acciones es la que debe ejecutarse en la situación actual. El problema aparece al considerar problemas complejos, donde el espacio de búsqueda de la solución es tan amplio que resulta imposible, en la práctica, alcanzar la solución mediante la ejecución, sin un plan previo, de distintas acciones básicas.

La arquitectura puramente reactiva no necesita de ningún otro nivel y la actuación final del robot se basa en la importancia para cada situación de la función que se puede ejecutar. A veces se define una relación entre las acciones básicas, de modo que una acción puede bloquear a otra e impedir su ejecución (Brooks, 1991). Otras veces se recurre a distintos algoritmos de fusión que permiten ponderar las salidas (Goodridge *et al.*, 1994) (Matellán *et al.*, 1995). Por ejemplo (Figura 2.5), si consideramos un robot móvil que tiene dos acciones posibles: evitar obstáculos y seguir un objeto, cuando la situación del entorno es tal que la función de evitar tiene que ser ejecutada, ésta bloquea automáticamente la acción de seguir.

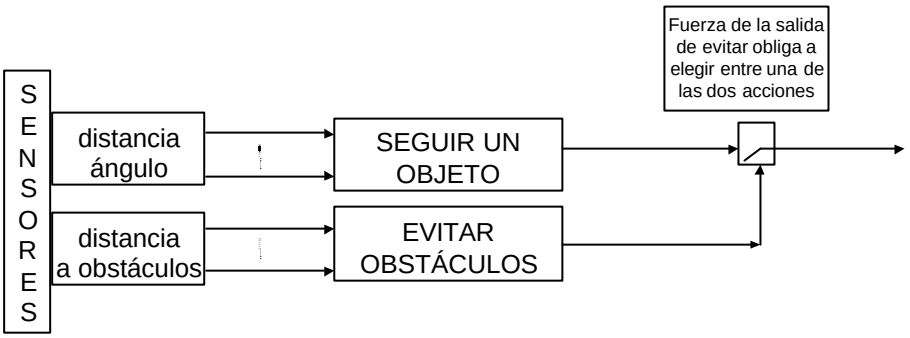

Figura 2.5. Ejemplo de Arquitectura Reactiva.

Para Brooks (1986) la mejor representación del entorno es el propio entorno, y el sistema de control debe razonar directamente sobre los datos adquiridos por el robot en ese momento. El concepto de reactividad define que el sistema de control debe utilizar únicamente la información actual para decidir la siguiente acción, tal y como muestra la Figura 2.6.

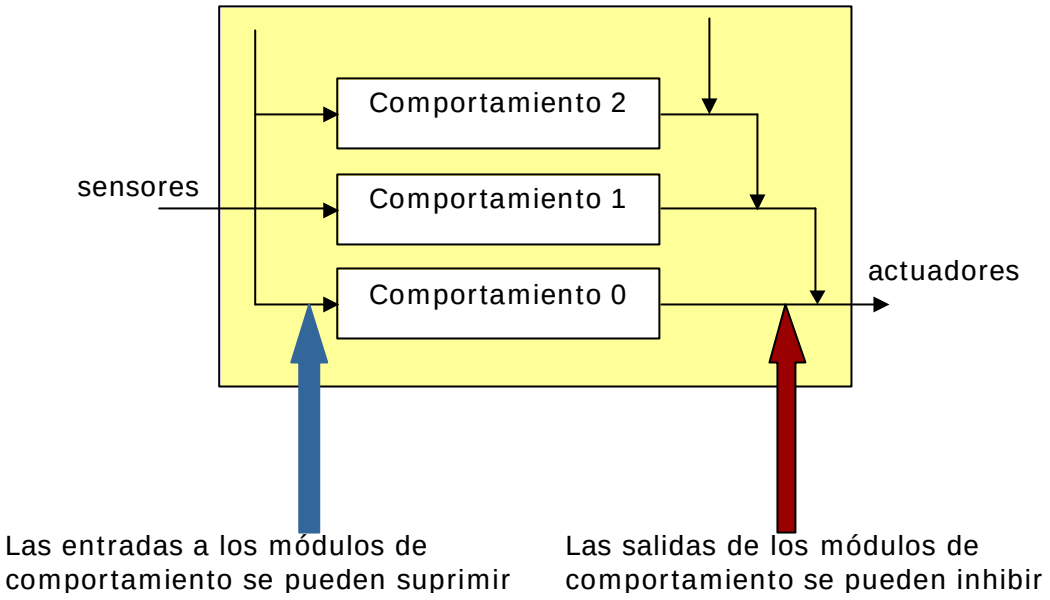

Figura 2.6. Arquitectura de subsunción (Brooks, 1986)

El concepto fundamental de la arquitectura reactiva (la acción básica o comportamiento), ha servido para el desarrollo de sistemas de aprendizaje utilizados para adaptar estas acciones tanto en arquitecturas reactivas como deliberativas (Berlanga *et al.*, 1999, 2002) (Matellán et al. 1998) (Molina et al. 200) y para optimizar los procesos de comunicación entre robots y la ejecución de acciones conjuntas (Matellán *et al.*, 1996).

2.2.3 Híbridas

Como hemos visto, tanto las arquitecturas reactivas como las deliberativas presentan ciertas limitaciones. Por ello, se han propuesto sistemas híbridos que pretenden combinar aspectos de ambos modelos. Una primera propuesta puede ser construir un agente compuesto de dos subsistemas: uno deliberativo, que utilice un modelo simbólico y que genere planes en el sentido expuesto anteriormente, y otro reactivo centrado en reaccionar a los eventos que tengan lugar en el entorno y que no requiera un mecanismo de razonamiento complejo.

Por su propia naturaleza estas arquitecturas son propicias para una estructuración por capas, que puede ser:

- *Vertical*: sólo una capa tiene acceso a los sensores y actuadores.
- *Horizontal*: todas las capas tienen acceso a los sensores y a los actuadores.

De la misma forma que las arquitecturas de subsunción, las capas se organizan jerárquicamente con información sobre el entorno a diferentes niveles de abstracción. La mayoría de las arquitecturas encuentran suficientes tres niveles:

- *Reactivo*, o de más bajo nivel. En él se toman decisiones acerca de lo que hacer en base a los estímulos recibidos del entorno en tiempo real. Suele estar implementado mediante arquitecturas de subsunción.
- *Conocimiento*, o nivel intermedio Se centra en el conocimiento que el agente posee del medio, normalmente con la ayuda de una representación simbólica del mismo.

- *Social*: es la capa de más alto nivel. En ella se manejan los aspectos sociales del entorno, incluyendo tanto información de otros agentes, como los deseos, intenciones, etc.

El comportamiento global del agente viene definido por la interacción entre estos niveles. Esta interacción cambia de una arquitectura a otra.

2.2.3.1 Ejemplos de arquitectura de capas

Uno de estos ejemplos es la arquitectura TOURINGMACHINES (Ferguson, 1992), donde cada nivel está continuamente sugiriendo qué acción realizar y existe un sistema de control para garantizar el correcto funcionamiento del agente (Figura 2.7). Se trata de una arquitectura de tres capas horizontales:

- Reactiva (respuesta inmediata a cambios del entorno, como reglas situación->acción, o entrada de sensor-> salida de actuador).
- Planificadora (comportamiento proactivo basado en esquemas de planes).
- Modeladora (modelo del mundo para anticipar conflictos y genera nuevos objetivos para la capa planificadora).

Todas ellas embebidas en un subsistema de control basado en reglas que pueden inhibir entradas y salidas, y determina qué capa tiene control sobre el agente.

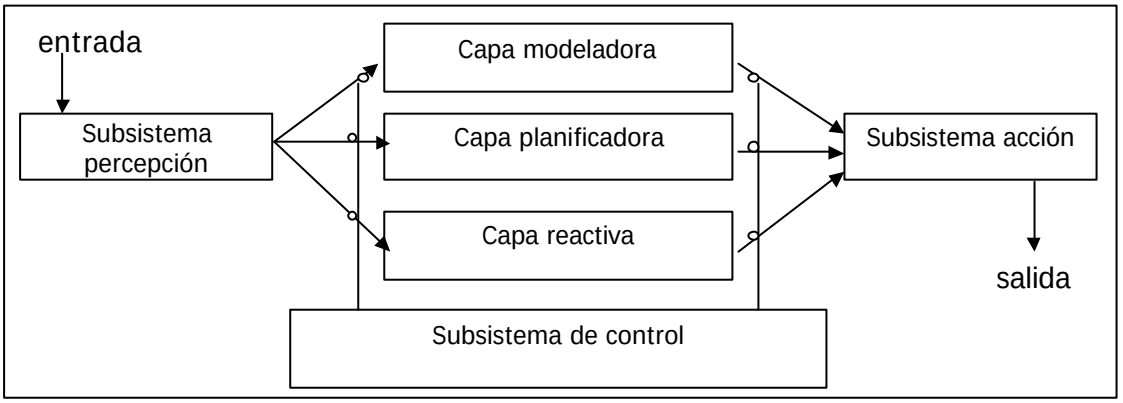

Figura 2.7 : Arquitectura de TouringMachines

Otra arquitectura como INTERRAP (Müller, 1997) plantea un modo de trabajo similar pero las capas están estructuradas verticalmente, incluyendo además una base de conocimiento también organizada en capas (Figura 2.7).

- Capa de comportamiento (reactivo): información del entorno.
- Capa de planificación (pro-activo): planes y acciones del agente.
- Capa de cooperación (interacción social): planes y acciones de otros agentes del entorno.

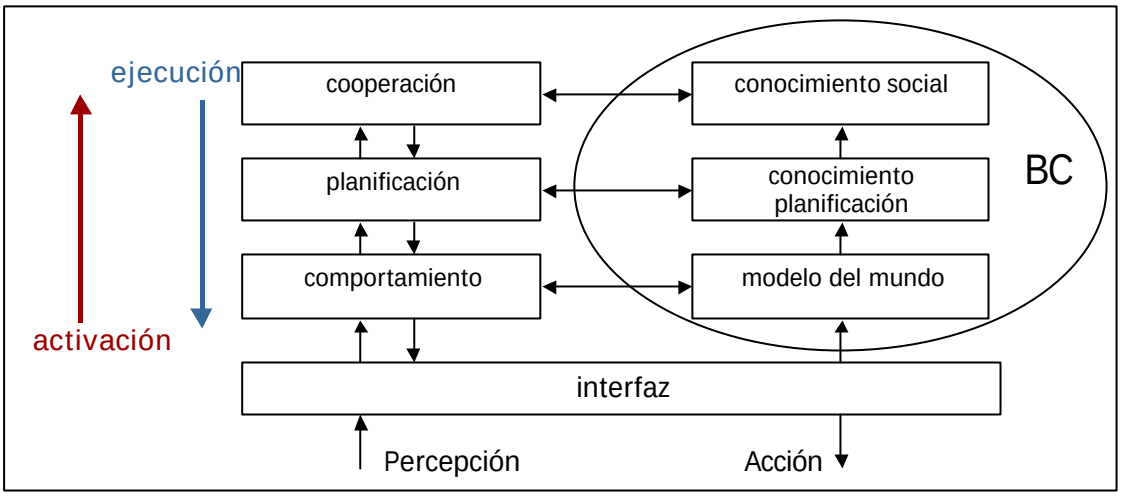

Figura 2.7. Arquitectura de capas verticales: InterRap

Este tipo de arquitecturas aparece en muchos problemas en los que es necesario integrar un razonamiento deliberativo para la negociación conducente a una modificación de la ejecución, pero donde la ejecución es independiente del razonamiento. Son sistemas de este tipo los sistemas de robótica donde se da una ejecución de acciones continúa y por lo tanto el bajo nivel es reactivo y al mismo tiempo se integran los robots en un sistema multiagente para la consecución de un objetivo conjunto. El mismo problema aparece con otros tipos de sistemas reales como los sistemas de vigilancia, y en particular en aplicaciones de radares multifunción de vigilancia área (Molina *et al*. 2002, 2003).

2.2.3.2 SISTEMAS INTELIGENTES ADAPTATIVOS

Un sistema inteligente adaptativo (Adaptative Intelligent System, AIS) es un sistema experto que razona e interactúa con su entorno en tiempo real (Hayes-Roth, 1988). En (Hayes-Roth, 1988) se muestra una arquitectura para construir tales agentes. La arquitectura consta de dos niveles: el físico, que efectúa los trabajos relacionados con la coordinación entre percepción y acción -tales como interpretar, filtrar y reaccionar al ambiente dinámico en el cual el agente está embebido- y el cognitivo, que recibe información del nivel físico, construye un modelo evolutivo del mismo e interpreta, razona y planifica a partir de él.

La arquitectura fue empleada para construir diversos agentes en dominios especializados, entre los que destaca GUARDIAN. Este sistema es capaz de monitorizar el estado de 20 variables de estado de cada paciente en una unidad de cuidados intensivos. GUARDIAN está formado por:

- un componente cognitivo
- un conjunto de subsistemas I/O asíncronos
- un conjunto de canales I/O
- un ciclo de razonamiento

Implementado como un sistema de pizarra, el componente cognitivo se comporta como un razonador de propósito general que realiza las funciones de un planificador dinámico. Éste es un proceso para la construcción y modificación incremental de planes, mediante el uso de patrones ordenados temporalmente para tareas de control de

decisiones, cada una de las cuales describe una clase de operaciones que el agente ejecuta en un intervalo de tiempo limitado. Las operaciones dentro del componente cognitivo son procesadas cíclicamente por turnos, siguiendo una agenda establecida, y un ejecutor de operaciones.

GUARDIAN emplea un ciclo de razonamiento con el fin de proporcionar la garantía para operaciones de tiempo-real. Esto se organiza a través del uso de heurísticas de control que tienen la misión de limitar el número total de operaciones que van a ser procesadas por el componente cognitivo.

2.2.3.3 Otros ejemplos

La arquitectura que presenta PHOENIX (Cohen, 1989) para el desarrollo de un agente propone cuatro componentes diferenciados: módulo cognitivo, módulo de acción refleja, sensores y actuadores..

La respuesta del sistema a un cambio en el entorno se genera a través de dos módulos: el primero, el de acción refleja, está compuesto de pares estímulo-respuesta y es capaz de calcular directamente una acción a partir de la llegada de un estímulo del entorno. Este módulo se utiliza cuando no se dispone de suficiente tiempo para que entre en acción el módulo de más alto nivel (módulo cognitivo). Este último es capaz de desarrollar respuestas más complejas, entre otras, planificación, coordinación, monitorización, etc.

Ogasawara diseña una arquitectura de varios niveles, alrededor de un núcleo principal que consiste en una búsqueda de soluciones mediante la maximización de la utilidad esperada de las acciones posibles (Ogasawara, 1991).

En un nivel superior, el sistema dispone de diferentes métodos de resolución de problemas, denominados por los autores arquitecturas de ejecución. Cada uno de los módulos maneja un tipo de conocimiento diferente y presenta diferentes soluciones. Un módulo de arbitraje elegirá, en función de la utilidad, cuál es la respuesta adecuada.

En un nivel inferior, dentro de cada uno de los módulos, la búsqueda de una solución se realiza en función de la situación del problema y del meta-conocimiento del que se dispone, guiándose también por la utilidad. Existe un módulo más elemental, basado en reglas de condición-acción, que proporciona respuestas inmediatas en función del estado situacional.

La arquitectura VCBP (Variational Calculus Based Planner) pretende reducir la carga de procesamiento que llevan asociados los sistemas de planificación en dominios dinámicos y continuos (Glez-Bedia *et al*., 2002; Glez-Bedia and Corchado, 2003). La carencia de modelos y técnicas propias para tratar eficientemente problemas de planificación sobre dominios continuos queda bien referenciada en (Salido and Barber, 2002). El motor inteligente que le proporciona autonomía al agente VCBP, le permite los dos tipos de capacidades que definen un agente híbrido (Castillo *et al*., 2003):

- capacidad de reacción frente a cambios inesperados. El agente tiene acceso en cualquier instante a un conjunto de planes posibles que se almacenan en una estructura CBR (Case based reasoning), y que evitan gastos para estructurar nuevos planes iniciales, facilitando las respuestas rápidas (Kolodner, 1983).

- capacidad de decisión deliberativa, utilizando un criterio de utilidad basado en la relación objetivos-recursos, que permite determinar cuál es el plan más adecuado para

ser ejecutado. El modelo variacional implementado estructura la gestión de datos del agente, y le permite identificar cuál es el plan mejor preparado para ser replanificado en el futuro si es que se ve interrumpido por sucesos no esperados, tal y como refleja la Figura 2.8. La estrategia variacional se ajusta adecuadamente al estudio de dominios continuos. La posibilidad de poder identificar, para cualquier instante temporal, el plan más robusto, en términos de su capacidad de replanificación futura, proporciona al agente condiciones útiles para enfrentarse a entornos sometidos a gran dinamismo.

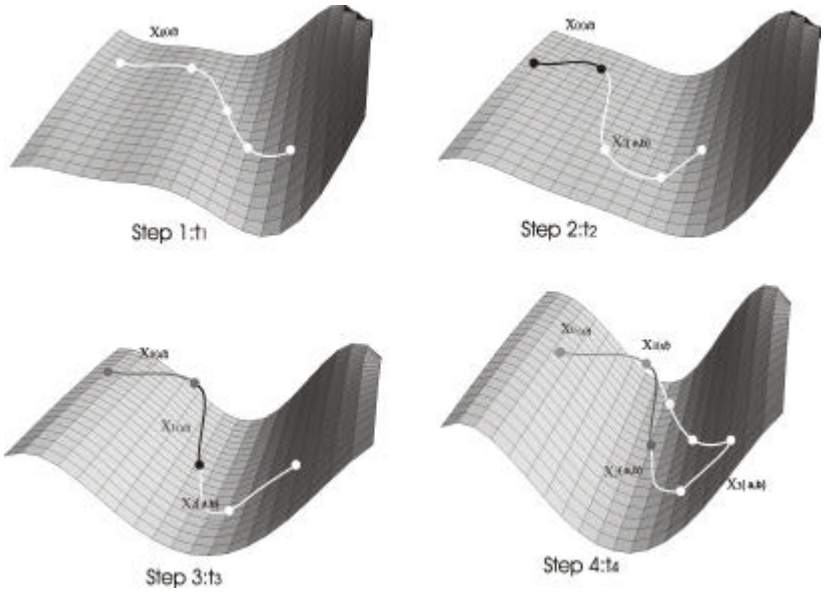

Figura 2.8. Evolución de un plan generado utilizando el planificador VCBP después de varias replanificaciones consecutivas (Glez-Bédia *et al*., 2002)

2.3 Arquitecturas Multiagente

La necesidad de desarrollar aplicaciones complejas compuestas de multitud de subsistemas que interaccionan entre sí obliga a distribuir la inteligencia entre diversos agentes y a construir *sistemas multiagentes*, que permiten la gestión inteligente de un sistema complejo, coordinando los distintos subsistemas que lo componen e integrando los objetivos particulares de cada subsistema en un objetivo común. Este tipo de sistemas, como se mencionó en el capítulo 1, se emplea en muchos casos: por ejemplo, cuando los problemas son físicamente distribuidos, cuando la complejidad de la solución requiere de experiencia muy heterogénea o cuando el problema a resolver está definido sobre una red de ordenadores. En realidad, la complejidad de la mayor parte de los problemas que nos encontramos hoy en día es tal que se requiere una solución distribuida, capaz de adaptarse a cambios en la estructura y en el entorno, así como una metodología de de desarrollo que permita la construcción de todo un sistema a partir de distintas unidades autónomas (Ferber, 1999).

En el capítulo 6 se estudiarán muchos casos reales de empleo exitoso de estos sistemas. Citemos ahora, por ejemplo, el sistema TravelPlan (Camacho *et al.*, 2001), capaz de generar planes de viajes a través de un sistema multiagente donde existen agentes especializados en modelado del usuario, en planificación y en búsqueda de información en la web (especialización funcional de los agentes). En otros casos, el sistema

multiagente puede que esté encargado de la resolución de un problema para lo cual cada uno de sus agentes implementa un algoritmo distinto, siendo el sistema multiagente el encargado de descubrir cuál es el agente que mejor resuelve el problema (Molina *et al*. 2001).

Si se dota a cada subsistema de una capacidad de decisión local, el problema de la gestión se puede abordar definiendo políticas de cooperación, coordinación y negociación entre agentes (todas estas cuestiones se abordarán con más profundidad en el capítulo 4). En este caso, es precisamente la colaboración entre los componentes del sistema la que puede permitir la resolución del problema y la consecución de los objetivos predefinidos (Cammarata *et al.*, 1988). En general, un sistema multiagente cooperante (Wesson *et al.*, 1988), presentará las siguientes características:

1. Estará formado por un conjunto de agentes, cada uno de los cuales mantiene sus propias habilidades: adquisición de datos, comunicación, planificación y actuación.
2. El sistema multiagente tiene una misión común. La misión puede descomponerse en diferentes tareas independientes, de forma que se puedan ejecutar en paralelo. El sistema multiagente debe ser capaz de asignar a cada uno de sus componentes una o varias tareas concretas teniendo en cuenta cual es el objetivo común.
3. Cada agente del sistema tiene un conocimiento limitado. Esta limitación puede ser tanto del conocimiento del entorno, como de la misión del grupo, como de las intenciones de los demás agentes a la hora de realizar sus propias tareas.
4. Cada agente del sistema tiene cierta especialización para realizar determinadas tareas, en función de lo que conoce, la capacidad de proceso y la habilidad requerida.

La distribución de las decisiones entre los distintos nodos del sistema permite fundamentalmente:

1. Mantener la autonomía de cada agente. Cada uno de los agentes decide en función de su propio entorno las tareas a llevar a cabo.
2. Eliminar la necesidad de que toda la información del sistema se encuentre en un único agente.
3. Descentralizar la toma de decisiones. Existen agentes que pueden tomar decisiones que involucran a todos los agentes e incluso planificar de manera coordinada un conjunto de tareas. El proceso de comunicación deberá atender a estas necesidades dotando de especial importancia a algunas de las decisiones de esos agentes centrales, pero la mayoría de las decisiones deberán ser consensuadas entre los agentes y no determinadas a priori. La descentralización hace que los sistemas sean más robustos ante posibles fallos.
4. Llevar a cabo acciones coordinadas. Para ello es necesario un mecanismo de comunicación entre los agentes.
5. Organizar dinámicamente la arquitectura de agentes, lo que permite la adaptación de la arquitectura a los cambios que se produzcan en el entorno en tiempo real.

En el problema de la coordinación entre los 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 identificación de la información que se debe compartir. El proceso de comunicación debe prefijar cómo y cuándo debe producirse la comunicación entre los nodos. La comunicación, el qué y el cómo, entre

los diversos agentes es muy importante a la hora de definir un sistema multiagente (los diversos aspectos de la comunicación entre agentes se abordarán en el capítulo 3). De aquí en adelante nos vamos a centrar en el análisis de una organización compuesta de agentes y por lo tanto deben identificarse sus elementos fundamentales. Ferber (1999) distingue estos tres métodos de análisis:

- Análisis funcional, que permite describir las diferentes funciones del sistema multiagente en sus diferentes dimensiones.
- Análisis estructural, que nos permite distinguir entre las distintas formas de organización e identificar los parámetros estructurales fundamentales.
- Análisis de parámetros concretos, centrado en las diferentes cuestiones que aparecen al pasar de una estructura abstracta a una organización concreta de agentes.

El análisis estructural intenta ordenar el conjunto de interacciones entre los agentes, analizando la relación abstracta que los relaciona y la manera en que esta relación evoluciona en el tiempo. Para realizar este análisis hay que concebir previamente el sistema multiagente desde el punto de vista de los agentes y las tareas que se deben realizar. Fundamentalmente se necesita conocer qué relación existe entre los distintos agentes y las tareas que debe realizar el sistema (la organización). Es decir, la relación existente entre la función productiva del agente y la de la organización.

Por una parte hay que identificar la relación entre un agente y el problema. La relación puede verse desde una perspectiva funcional (qué funciones realiza el agente y qué funciones debe llevar a cabo el sistema), desde la perspectiva de la distribución (los agentes están distribuidos espacialmente, encargándose cada uno de ellos de una determinada parte del problema) y, por último, se puede analizar desde una perspectiva orientada a objetos (considerando cada agente como un sistema que recibe mensajes y entrega productos ya terminados).

Por otra parte es necesario identificar la relación entre los agentes y las diversas tareas que deben llevarse a cabo. Es decir, cómo se distribuyen las tareas entre los agentes y cuál es la redundancia que existe (cuántos agentes pueden llevar a cabo qué tareas). En este caso existen dos perspectivas, una vertical, donde se analizan las funciones y su sucesiva descomposición y cómo se distribuyen estas tareas; y otra horizontal, en la que se describen las funciones que son compartidas por los agentes. Por ejemplo, en una sociedad de robots que manipulan objetos, el análisis vertical identifica cada una de las funciones de manipulación y las sucesivas sub-funciones de estas, mientras que el análisis vertical analiza las tareas de percepción, actuación y razonamiento que deben tener los robots para llevar a cabo las distintas funciones específicas. En resumen, el análisis vertical y horizontal son complementarios y cada uno de ellos se centra en una forma distinta de describir las tareas: la vertical, analizando la descomposición de tareas; y la horizontal, describiendo las características de las tareas con parámetros comunes a todas las funciones que deben llevarse a cabo.

Una vez que se tiene una descripción del sistema multiagente es posible analizar las relaciones abstractas entre los agentes que existen en la organización, los modos de enlace de los agentes y las subordinaciones que existen entre los agentes para la toma de decisiones.

Las relaciones abstractas describen las formas de interacción entre las distintas clases de agentes, es decir entre roles. Estas relaciones son:

- Relación de conocidos. Esta relación indica que dados dos agentes, uno tiene una representación del otro y conoce su dirección. Está es la relación mínima entre dos agentes y es el soporte de otras relaciones.
- Relación de comunicación. Esta relación indica que un agente puede mandar mensajes a otro. La relación de conocidos es el soporte para el canal de comunicación.
- Relación de subordinación. Esta relación describe la transferencia de una ejecución (tarea) entre dos agentes.
- Relación operativa. Esta relación representa las dependencias ente tareas relacionadas. Es decir puede ocurrir que para que un agente ejecute una tarea sea necesario que otro agente haya ejecutado previamente otra.
- Relación de información. Esta relación establece la validez de las dependencias ente las cosas que un agente conoce. Esto sucede cuando el conocimiento de un agente depende de otro agente, y por lo tanto el conocimiento del primer agente está fundamentado en la confianza que deposita en el otro agente.
- Relación de conflicto. Esta relación indica que los agentes tienen un conflicto por el acceso a los recursos y que será necesario coordinar las tareas mediante una negociación.
- Relación competitiva. Esta relación se corresponde con una competición entre los agentes y es una señal de que sus objetivos son incompatibles.

Las relaciones abstractas además pueden ser estáticas o dinámicas. Las primeras se determinan al definir la organización y no cambian con la ejecución mientras que las segundas cambian en función de la ejecución. La caracterización de estática o dinámica únicamente tiene sentido en tres casos:

- Relación de subordinación estática o dinámica. En la primera se define qué agente está subordinado a otro por lo que se obtiene una relación maestro-esclavo. En la segunda se tiene una situación de demanda de servicios por parte de un agente, es decir está basada en la petición por parte de un agente. Dicha petición puede ser aceptada o no por otro agente lo que implica que la relación depende de la situación del sistema en cada momento.
- Relación operativa estática o dinámica. La relación estática define las dependencias de las tareas desde un principio. La dinámica distribuye las tareas entre los agentes en función de su disponibilidad. Básicamente la ejecución de una tarea dependiente está íntimamente ligada a que otro agente se haya comprometido a la ejecución de la tarea de la que depende.
- Relación de información estática o dinámica. En el primer caso existe una relación de confianza mutua definida estructuralmente. En la relación dinámica el agente está obligado a demostrar la validez de la información y por lo tanto la relación entre los agentes puede modificarse en función de cómo se modifique la confianza en el agente.

Los modos de enlace definen cómo están unidos los agentes entre sí y la propia capacidad del agente para automodificarse (cambiar sus roles). Estos enlaces pueden ser fijos, variables o evolutivos. Los enlaces fijos impiden la reorganización y la adaptación al entorno de la organización y del agente convirtiendo al sistema en un programa, por lo que no son útiles para un sistema multiagente. Los enlaces variables permiten una

adaptación controlada de los agentes en alguno de sus parámetros, este tipo de adaptación es la más utilizada en sistemas multiagente. Por último la evolutiva permite que las relaciones y los propios agentes evolucionen sin definir los límites de la misma.

Las estructuras jerárquica e igualitaria definen la relación de subordinación. En la primera algunos agentes tienen la capacidad de ordenar a los demás agentes de manera que controlan y coordinan todas las actividades. En la segunda todos los agentes participan de la misma manera en la toma de decisión final, si el enlace es fijo estos sistemas se utilizan como sistemas de distribución de tareas. Al diseñar los agentes se puede definir la estructura de la organización a priori o a posteriori cuando emerge durante la propia ejecución del sistema.

Además del análisis llevado a cabo desde un punto de vista funcional y estructural queda por analizar cómo distribuir las habilidades entre los agentes, lo cual depende fundamentalmente del tipo de problema que se pretenda resolver. Los casos extremos son: que todos los agentes tengan todas las habilidades posibles o que cada agente tenga una única habilidad (sean especialistas). Para analizar esta distribución se pueden utilizar dos índices el grado de especialización y el grado de redundancia. El primero evalúa la relación existente entre el número de habilidades que tiene el agente para un problema dado y el número de habilidades existente para ese problema. El segundo evalúa la relación existente entre el número de agentes que tienen la misma habilidad y el número de agentes. A partir de estos parámetros puede identificarse cuatro puntos extremos:

- Organización no redundante y hiperespecializada. Cada agente sólo tiene una habilidad y nadie más que ese agente la posee. Esta organización es típica de una descomposición funcional, donde cada función es un agente.
- Organización redundante y especializada. Cada agente sólo tiene una habilidad y todos los agentes la poseen. Este tipo de organización sirve para optimizar algunos problemas de ejecución cuando se utilizan varias máquinas para resolverlos.
- Organización redundante y general. Cada agente tiene varias habilidades y varios agentes las poseen. Esta es la organización típica de los sistemas multiagente.
- Organización no redundante y general. Cada agente tiene todas las habilidades y nadie más que ese agente las posee. En este caso el sistema multiagente pasa a convertirse en un agente.

2.4 La arquitectura FIPA

Como ocurre con toda nueva tecnología, uno de los problemas que hay que resolver para facilitar su aplicabilidad es la interoperabilidad (facilidad de interconexión e integración de sistemas basados en dicha tecnología) y la apertura (posibilidad de extensión). Para ello es importante disponer de estándares y en el caso de los agentes la organización que más ha trabajado en este sentido es FIPA (Foundation for Intelligent Physical Agents). Entre las especificaciones de FIPA se definen características que deben cumplir las plataformas de gestión de sistemas multiagentes. Aparte de implementaciones originalmente concebidas a partir de este estándar (como FIPA OS y JADE), actualmente la mayoría de los entornos de desarrollo y ejecución de agentes tienden a adoptar, o ser compatibles, con FIPA.

A la hora de definir la plataforma de agentes, FIPA ha seguido el principio de definir solo el comportamiento externo (interfaz), dejando las decisiones de diseño a cada equipo de desarrollo. Otro principio que guía todo el estándar es conseguir un sistema totalmente abierto de tal forma que sistemas heterogéneos puedan interactuar a nivel de sociedades de agentes. El modelo FIPA establece el modelo lógico referente a la creación, destrucción, registro, localización y comunicación de agentes.

La Plataforma de Agentes (*Agent Platform*), como núcleo del modelo de referencia de FIPA, proporciona la infraestructura para el desarrollo y uso de agentes. Esta plataforma contiene todos los recursos hardware y software (sistema operativo, software de comunicaciones, middleware y software de gestión de agentes) necesarios para poner en marcha esta infraestructura. La idea es que en cada implementación concreta se tomen las decisiones relativas a las componentes usadas para desarrollarla, pero manteniendo siempre un conjunto de interfaces externas que permitan la interoperabilidad entre plataformas (por ejemplo para comunicación y gestión de agentes).

El estándar FIPA define los servicios que debe proporcionar toda plataforma de agentes: un sistema encargado del transporte de mensajes (*Internal Platform Message Transport*), un sistema de gestión de agentes (*Agent Management System*), un servicio de directorio (*Directory Facilitator)* y un canal de comunicaciones para los agentes (*Agent Communication Chanel*). Cada uno de estos servicios, excepto el de transporte de mensajes, es suministrado por agentes especializados, lo cual supone que la comunicación con ellos será mediante mensajes ACL (*Agent Communication Language,* que se describe en el capítulo de comunicación de agentes) mediante la ontología definida para ese servicio.

La Figura 2.9 muestra el conjunto de componentes de una plataforma de agentes FIPA:

- Agent Management System (AMS). El AMS es el elemento de gestión principal, que conoce en todo momento el estado de su plataforma y de los agentes que pertenecen a ella. Entre sus responsabilidades (servicios que ofrece) están la creación, destrucción y control del cambio de estado de los agentes, supervisión de los permisos para que nuevos agentes se registren en la plataforma, control de la movilidad de los agentes, gestión de los recursos compartidos y gestión del canal de comunicación. Cada AMS proporciona también un servicio de nombres (*Agent Name Service-ANS)*, también llamado servicio de *páginas blancas*. Este servicio asocia el nombre que identifica a un agente en su sociedad y la dirección de transporte real en la que se encuentra. De tal forma que proporciona un método básico para buscar un agente dentro de la plataforma. Como elemento de gestión, el AMS permite controlar el ciclo de vida de los agentes. Cada agente estará en un uno de los siguientes estados: Iniciado, Activo, Suspendido y Esperando.
- Directory Facilitator (DF). Como complemento al Servicio de Nombres, la plataforma de agentes incluye un servicio de *páginas amarillas*, que permite buscar un agente por sus capacidades y no sólo por su nombre. Esta labor la realiza el *Directory Facilitator* (DF). Los agentes se registran en él indicando los servicios que ofrecen. Cuando otro agente tiene unas necesidades concretas lanza una búsqueda del servicio deseado obteniendo los agentes que le ofrecen estos servicios.
- Agent Communication Channel (ACC). Todos los agentes FIPA deben tener acceso a un ACC. El ACC es el elemento encargado de gestionar el envío de mensajes entre agentes de una plataforma y entre agentes de distintas plataformas. Esta gestión consiste principalmente en el encaminamiento de los mensajes ACL desde su agente origen a su agente destino. Cuando un agente en una plataforma quiere enviar un mensaje a otro en una localización remota, entrega este mensaje al ACC

que se encargará de entregárselo al ACC de la plataforma donde se encuentre el otro agente. Este último ACC notificaría al agente destino que ha recibido un mensaje. El modelo de comunicación entre agentes es asíncrono, lo que implica que: el ACC no se queda bloqueado ante el envío o recepción de mensajes, existen colas de envío y recepción de mensajes, para las cuales deben definirse políticas de gestión de colas de mensajes.

- Internal and Agent Platform Message Transport (IPMT-APMT). El Internal Platform Message Transport (IPMT) representa toda la infraestructura de comunicaciones que va a permitir que dos agentes dentro de la misma plataforma puedan comunicarse. El Agent Platform Message Transport (APMT) es otro elemento que hace la misma labor que el IPMT pero a nivel de comunicaciones entre plataformas (ACCs).

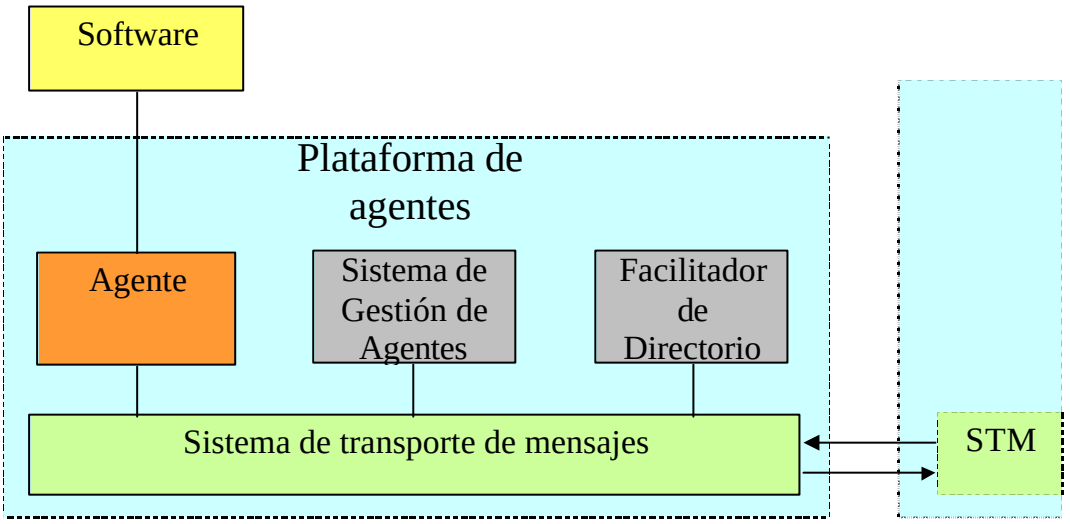

Figura 2.9. Modelo de referencia de FIPA

2.4.1 Seguridad en el desarrollo de Sistemas Multiagente

Los problemas de seguridad son inherentes a cualquier sistema distribuido y más aún en sistemas multiagente, debido a las características de autonomía y el alto grado de colaboración y cooperación que pueden existir entre los distintos agentes. Por ello FIPA ha estudiado soluciones de seguridad para que los sistemas multiagentes puedan desplegarse en sistemas abiertos con las garantías requeridas. Este tema no está cerrado actualmente y es un área de investigación permanente..

Las recomendaciones de FIPA están basadas en la utilización de los estándares existentes siempre que esto sea posible. Por ejemplo, se plantea el uso del estándar de seguridad X.509 basado en infraestructuras de seguridad de clave pública (PKI). El estándar X.509 presenta un modelo de seguridad asimétrico o basado en certificados de clave pública y clave privada. Para cada entidad se definen dos tipos de claves, la clave privada a las que solo tiene acceso la entidad propietaria del certificado, y la clave pública a la que tienen acceso todas las demás entidades de la sociedad. Por cada entidad, la autoridad certificadora correspondiente genera un par de claves (pública y privada) para el cifrado (confidencialidad) y otro par de claves (pública y privada) para la firma (autenticidad). Cada entidad recibe sus claves privadas por parte de la autoridad certificadora. Las claves públicas de todas las entidades están guardadas en

una estructura de directorio (LDAP) y pueden ser consultadas en cualquier momento. También es posible que una entidad distribuya sus claves públicas a las entidades con las que quiere comunicarse de forma segura.

Este modelo de seguridad se presenta a nivel de agentes. Son posibles otros modelos de seguridad de más bajo nivel como el uso de sockets seguros, SSL (*Secure Socket Layer*), a nivel de la capa de transporte.

Otros problemas que pueden surgir en la comunicación entre agentes o su despliegue, y que contempla FIPA son los siguientes:

- **Copia de Código y datos**. Los sistemas de agentes se suelen implementar usando lenguajes interpretados como Java o TCL para conseguir una mayor portabilidad e interoperabilidad. Pero esto supone que el código y los datos usados son fácilmente accesibles y reproducibles.
- **Alteración de código y datos**. De forma similar a la copia, la alteración de código y datos de un agente puede ser un problema potencial. Una plataforma de agentes puede estar preparada para atacar a los agentes que lleguen a ella.
- **Ataques en grupo**. Aunque una plataforma este protegida ante el ataque de algún agente con malas intenciones, son posibles otros tipos de ataque en grupo complicados de evitar.
- **Copia y reenvío**. Un mensaje puede ser copiado y enviado a otros agentes no incluidos en la lista de destinos del mismo.
- **Denegación del servicio**. Una plataforma de agentes puede negar ciertos servicios a un agente sin motivo aparente.
- **Rechazo de responsabilidad**. Es muy complicado repartir responsabilidades cuando ocurre un problema. Una plataforma o agente puede actuar de forma maliciosa y no admitir su responsabilidad.
- **Mascarada**. Un agente puede hacerse pasar por otro en una conversación para obtener información o para acceder a servicios que de otra forma no tendría accesibles.
- **Liquidez**. En algunos entornos se han creado sistemas de pago (tickets) para que un agente tenga que pagar para conseguir un servicio. Esto implica que se debe gestionar la liquidez de los agentes y que el proveedor del servicio ofrezca tal servicio una vez cobrado.

2.4.2 Movilidad de Agentes en la Plataforma

La tecnología de agentes móviles se puede considerar un campo de investigación que ha seguido un desarrollo paralelo al de los agentes inteligentes. Hasta ahora se estaban haciendo esfuerzos por separado pues ambas tecnologías estaban poco maduras. Ahora parece que es posible integrar ambas partes para conseguir un modelo de agentes que aporte todas las capacidades deseables.

Siguiendo la filosofía del estándar, FIPA no propone soluciones tecnológicas sino más bien aporta ideas acerca de cómo integrar de forma homogénea y natural las capacidades de los agentes móviles dentro de la plataforma. La interoperabilidad entre sistemas es muy importante en este campo, pues si dos sistemas de distinto origen no pudiesen intercambiar agentes la movilidad no sería aprovechable. Cuando se trata de agentes móviles podemos trabajar con dos soluciones, más o menos complejas de llevar a cabo:

- **Migración**: Este modelo es más complejo, pues requiere la transferencia de código y datos del agente a la plataforma remota. Cuando un agente va a migrar se

"suspende", se transfiere por la red y se "despierta" en el otro extremo prosiguiendo la ejecución de sus tareas en el mismo punto donde las había dejado.

- **Clonación**: En este caso se crea una copia del agente en el nodo remoto que se encarga de hacer el trabajo solicitado por su clon en la máquina origen. En este caso no hay una transferencia efectiva de código y datos, sino solo un conjunto de ordenes que el clon debe realizar.

Como para otros servicios prestados en la plataforma de agentes, el estándar define un conjunto de mensajes ACL junto con la Ontología usada para los mismos (ver *FIPA Specification – Part 11, Agent Management support for Mobility*). Un ejemplo es este mensaje con el que un agente solicita a su AMS la migración a una dirección remota.

```
(request
  :sender an-agent@async://fipa97.org/acc
  :receiver an-ams@async://fipa98.org/acc
  :content
    (action an-ams@async://fipa98.org/acc
      (move
        (:agent-name an-agent@async://fipa97.org/acc)
        (:address async://fipa97.org/acc)
        (:destination an-ams@async://fipa98.org/acc)
        (:agent-profile ...)
        (:agent-mobility-protocol "...")
        (:agent-code "...")
        (:agent-data "...")
        (:agent-version "...")
  :protocol fipa-request
  :ontology fipa-mobile-agent-management
)
```

FIPA define dos modelos de movilidad en función de quién tenga que hacer la gestión de la transferencia:

- **Modelo Simple**: El AMS (*Agent Management System*) es el responsable de realizar toda la gestión necesaria. El agente solicita a su AMS la transferencia y éste se ocupa de llevarla a cabo.
- **Modelo Complejo**: El agente es el responsable de realizar todas las operaciones que le permiten desplazarse a la plataforma remota.

Conseguir la movilidad dentro de un sistema de agentes no es nada sencillo. Hay algunos problemas que todavía no se han resuelto de una manera general, y hasta que no se consiga no podremos pensar en un sistema comercial basado en agentes móviles como los que presenta FIPA en su estándar. Algunos de estos problemas están relacionados con la seguridad, el control del estado de los agentes, la integridad del proceso de transferencia, la continuidad de las comunicaciones de los agentes móviles, los requisitos de recursos en cada lugar y la gestión de fallos.

2.4.3 Servicio de Ontología

Toda la plataforma FIPA se basa en la comunicación entre agentes mediante ACL (*Agent Communication Language*, descrito en el capítulo 3). Un mensaje ACL consiste básicamente en una expresión de un determinado lenguaje y con un conjunto de términos específicos de la ontología usada. Es decir, cuando dos agentes están conversando es necesario que ambos estén de acuerdo en el lenguaje y la ontología empleados. Una Ontología es un conjunto de símbolos o términos junto su correspondiente interpretación o significado. Cuando dos agentes usan una misma Ontología se aseguran que sus interpretaciones de lo que se está diciendo coinciden.

Cuando un agente se registra en el DF (*Directory Facilitator*) informa de las ontologías de las que tiene conocimiento, de tal forma que, cuando otro agente quiere conversar con él utilizará una de ellas a fin de conseguir un entendimiento mutuo. En un dominio

pequeño es posible que las ontologías aplicables se incluyan en el propio código del agente. De esta forma, estas ontologías serían cerradas y estarían limitadas al contexto en el que van a ser utilizadas.

Cuando pensamos en sistemas de mayor envergadura no hay más remedio que utilizar Servicios de Ontología (*Ontology Agent*), de tal forma que todos los agentes puedan utilizar dichos servicios para obtener toda la información necesaria.

Las labores básicas del Servicio de Ontología son:

- Mantiene un conjunto de ontologías de uso público accesible a los agentes.
- Traduce expresiones entre diferentes ontologías.
- Responde a consultas sobre términos de las ontologías que gestiona.
- Facilita la identificación y uso de ontologías compartidas entre los agentes.
- Descubre nuevas ontologías y las pone a disposición de todos los agentes.

En la actualidad existen una serie de servidores de ontologías fuera del ámbito de FIPA, tales como el servidor de Ontolingua, el servidor de XML/RDF y el servidor ODL a los que se accede a través de APIs como OKBC, HTTP y ODL respectivamente. FIPA no pretende sustituir estas implementaciones, sino agruparlas de forma que se puedan utilizar de forma transparente dentro de la plataforma.

Como hemos dicho, el *Ontology Agent* tiene como principal objetivo mostrar a los agentes FIPA una visión unificada de los servicios de ontología ofrecidos por los distintos servidores existentes.

2.4.4 Transporte de mensajes de la plataforma de agentes

Como se ha mencionado anteriormente el APMT (*Agent Platform Message Transport*) es el encargado de realizar la transferencia de mensajes ACL entre las ACCs (*Agent Communication Channel*) de dos plataformas. Aunque en un principio la labor de diseño de esta componente de bajo nivel se dejó en manos de cada grupo de trabajo que implementase la plataforma FIPA, finalmente se decidió introducir dentro del propio estándar debido a la gran importancia que tiene dentro del sistema. Para describir este componente, se introduce la siguiente terminología:

- AMTS (*Agent Message Transport Service*): Es el objeto que implementa el servicio de transporte de mensajes en un AP. Cada agente de la plataforma puede conseguir una referencia al AMTS a través de AMS (*Agent Management System*).
- *Conversation Factory* : Es una interfaz implementada por el AMTS que permite a un agente establecer un contexto de comunicación (*Conversation Context*). Cuando un agente desea empezar una conversación con otro necesita obtener un nuevo contexto.
- *Conversation Context*: Representa un extremo lógico de un canal de comunicación establecido para el intercambio de mensajes ACL. Define la información que necesita mantener el sistema para lograr un flujo de mensajes entre dos agentes. Un agente tendrá tantos objetos de este tipo como número de conversaciones en el que esté implicado. De forma muy básica se puede decir que un contexto de comunicación está definido por el origen, el destino, un método de transporte y la calidad del servicio prestado. Además, cada objeto de esta clase contendrá una cola de mensajes en espera de ser recibidos por el agente y una función de *callback* mediante la que se notifica al agente de la llegada de nuevos mensajes pertenecientes a dicha conversación.

- *Transport Factory*: Es una interfaz implementada por el AMTS que permite a un agente investigar el conjunto de mecanismos de transporte soportados (iiop, http, wap,etc.) y conocer algunas de las características de los mismos.
- *Transport*: Representa un mecanismo de transporte de mensajes concreto. Cada vez que un agente desea establecer una nueva conversación tiene que identificar el método de transporte a usar o bien delegar esta responsabilidad al propio AMTS.
- *Address*: Representa un extremo del canal de transporte utilizado. Dependiendo de del tipo de transporte elegido las direcciones serán diferentes (direcciones IP, URL, RFC_822 email, etc.).
- *Destination*: Identifica la entidad remota con la que se está produciendo la comunicación o intercambio de mensajes. Puede haber un solo destino asociado con varias direcciones correspondientes a diferentes tipos de transporte. Este modelo se basa en la vida real, en la que una persona puede tener varias direcciones de contacto tales como su teléfono personal, teléfono móvil, fax, dirección postal, etc.
- *Quality of Service*: Encapsula el conjunto de propiedades que gobiernan el modo en que el canal de transporte realizará el intercambio de mensajes entre los extremos de la comunicación. Algunas de estas propiedades son la integridad de datos, la privacidad, el rendimiento mínimo y el ancho de banda mínimo. Esto permitirá elegir en cada momento la calidad de servicio que se desea contratar de acuerdo con las necesidades.
- *Message*: Es la unidad básica de información que se va transmitir por el canal de comunicación. Como ya se ha explicado, esta unidad va a ser el mensaje ACL.

Una vez que se tienen todos estos componentes disponibles en el sistema el modelo de comunicación básico sería el siguiente:

1. Un agente A empieza su ejecución en una plataforma de agentes.
2. El agente A se registra como un participante potencial de conversaciones.
3. Un agente B empieza su ejecución en otra plataforma de agentes.
4. El agente B se registra como un participante potencial de conversaciones.
5. El agente B identifica a A como un agente capaz de conversar.
6. El agente B crea un contexto de comunicación para conversar con A.
7. El agente B envía un mensaje al agente A.
8. El AMTS de la plataforma de agentes de A recibe el mensaje enviado por B.
9. Un contexto de comunicación es establecido por A para comunicarse con B.
10. El mensaje de B es enviado a A mediante la función de callback establecida.
11. El agente A contesta al mensaje del agente B.
12. Se produce un intercambio de mensajes.
13. El agente A finaliza su comunicación con el agente B.
14. El agente B es notificado de la finalización de la comunicación.

2.4.5 Nombrado de agentes

En un principio, FIPA fijó un modelo de nombrado de agentes estático donde cada agente tenía un identificador único (GUID) que tenía la siguiente estructura:

<name>@<hostname>:<port>/<target>

siendo:

<u>Name</u> : identificador del agente en la plataforma de agentes a la que pertenece.

<u>HostName</u>: Dirección IP (o nombre DNS) del host en el cual se está ejecutando su ACC.

Port: Puerto en el que escucha el ACC.

Target: Identificador del agente que debería recibir el mensaje para encaminarlo al agente destino. Normalmente será el propio ACC.

Un ejemplo de GUID podría ser: *acc@iiop://www.ucm.es:50/acc*

Posteriormente se decidió que la estructura del nombre debía ser más flexible y completa para acometer futuras necesidades.

AgentName= (":guid" Word NameFields*)

NameFields= ":addresses" "(" CommAddress+ ")"

| ":resolver" "(" AgetName+ ")"

| ":authenticator" "(" AgentName+ ")"

donde:

GUID: Identificador único del agente en la plataforma de agentes. No tiene porque contener la dirección de transporte como en el modelo simple.

Addresses: Dirección/es de transporte en la que se encuentra el agente.

Resolver: Dirección donde se encuentra el Servidor de Nombres (ahora en el AMS)

Authenticator: Dirección del agente de autentificación (ahora el propio AMS).

Un ejemplo podría ser:

:guid MiAgente@iiop://www.ucm.es

:resolver (

:guid ams@www.ucm.es

:addresses (iiop://www.ucm.es:81/ams

http://www.ucm.es:80/ams)

)

:addresses (iiop://fipa.org/MiAgente)

:authenticator (iiop://fipa.org/Autentificación)

Este modelo está basado en pares atributo-valor y por tanto puede ser extendido de forma sencilla para recoger nuevas necesidades que pudiesen aparecer en el futuro.

2.5 RETSINA: una arquitectura multiagente

RETSINA (Reusable Environment for Task-Structured Intelligent Network Agents) es una arquitectura multiagente independiente de dominio, que puede aplicarse para el desarrollo de agentes cooperativos y que se ha desarrollado en el *Robotics Institute, Carnegie Mellon University.* Esta arquitectura se desarrolló para solucionar problemas reales de recolección de información distribuida y se ha utilizado con éxito en problemas relacionados con la toma de decisiones en organizaciones empresariales y en

el manejo de carteras financieras. RETSINA se caracteriza por permitir el desarrollo de SMA en los cuales los agentes operan asíncronamente, y colaboran entre ellos y con sus usuarios, tal y como se muestra en la Figura 2.10. Estos agentes buscan activamente la información y cooperan para realizar tareas de recolección e integración de información. Además, soporta comunidades de agentes heterogéneos. RETSINA se ha desarrollado sobre la base de que los agentes forman una comunidad en la que todos los agentes están al mismo nivel y sin imponer restricciones, a nivel de infraestructura, en las comunicaciones. La infraestructura distribuida de servicios que proporciona esta arquitectura impide la existencia de un control centralizado.

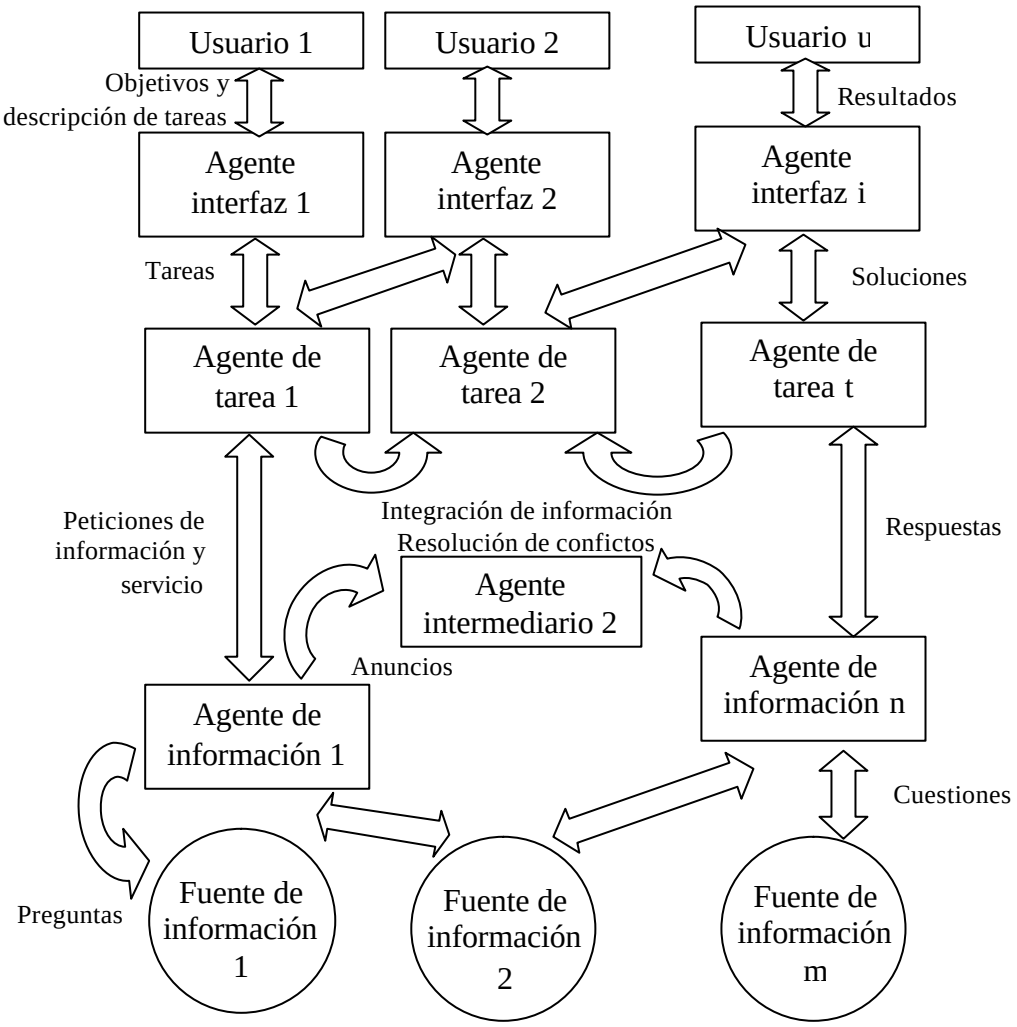

Figura 2.10. Arquitectura RETSINA

RETSINA se ha utilizado, por ejemplo, para el desarrollo un sistema de ayuda para la eliminación de minas mediante robots, llamado AgentStorm. Los agentes de eliminación de minas interactúan para encontrar rutas, facilitando la eliminación de campos de minas. AgentStorm permite la experimentación con distintas estrategias de cooperación multi-robot y de comunicación, de manera que cada robot planifica su

estrategia en base a suposiciones sobre lo que están realizando sus compañeros y sobre datos sensoriales (Sycara *et al*., 1996).

La Figura 2.10 muestra los distintos tipos de agentes que se pueden integrar en esta arquitectura: Agentes de interfaz, Agentes de tareas, Agentes de información y Agentes intermediarios.

Los agentes de interfaz interactúan con el usuario y hacen uso de su conocimiento sobre el dominio, del papel que el usuario juega en el contexto del problema y de la funcionalidad que espera conseguir, para personalizar su interfaz y adaptarse al usuario. Estos agentes están dotados de la capacidad de anticiparse a las necesidades del usuario utilizando información histórica relacionada con el usuario. Estos agentes actúan como interlocutores con el sistema ocultando los aspectos más complejos.

Los agentes de tarea tienen capacidad de planificación, para lo cual utilizan su conocimiento sobre el dominio. Estos agentes son capaces de descomponer una petición de información de alto nivel en tareas de menor nivel y son capaces de generar planes de ejecución de las tareas identificadas, así como de coordinar su ejecución. Las tareas que llevan a cabo estos agentes son:

- Recibir las especificaciones de una tarea delegada que le envía un agente de interfaz,
- Interpretar las especificaciones e identificar los objetivos que deba alcanzar el sistema,
- Formular planes para satisfacer esos objetivos,
- Identificar los subobjetivos del plan, y
- Descomponer el plan y llevar a cabo la coordinación e intercambio de información con los agentes adecuados para la ejecución del plan, monitorización y consolidación de los resultados.

La planificación debe adaptarse al estado actual del sistema así como tener en cuenta potenciales cambios de recursos, ya que en función del problema al que se aplique puede que algunos recursos de información no estén disponibles en algunos periodos de tiempo. Por ejemplo, en el caso de AgentStorm, es posible que en la batalla se pierdan líneas de comunicación.

Los agentes de información proporcionan acceso inteligente a sistemas heterogéneos de información, actuando como un *wrapper* o intermediario, o incluso como un *gateway* para las fuentes de información. Estos agentes disponen de modelos de los recursos de información asociados y de estrategias para la selección de las fuentes de información, acceso, resolución de conflictos, y fusión de la información. Los agentes de información pueden recibir consultas de datos relacionados con las fuentes de información asociadas. Estas consultas pueden ser periódicas. Además pueden monitorizar la información procedente de las fuentes, y recibir notificaciones relacionadas con el cumplimiento de condiciones.

Los Agentes intermediarios se encargan de poner con contacto a los agentes que necesitan un servicio, con los que lo proporcionan. Analizan semánticamente la información que requiere el agente y devuelven los recursos disponibles para poder ser utilizados. Pueden ser de varios tipos:

- **Agent Name Service**: Resuelve la localización de un agente utilizando páginas blancas.
- **Matchmakers**: Negociadores que se encargan de emparejar los agentes, realizando búsquedas por en páginas amarillas.

- **Agentes blackboard**: Coleccionan demandas de servicios.
- **Brokers** : Procesan los anteriores
- **MAS Interoperator**: Conectan RETSINA MAS con otros sistemas multiagente externos.

La tabla 2.2 muestra las competencias y funciones de los componentes, agentes y módulos de los agentes de la arquitectura RETSINA:

Infraestructura RETSINA MAS	**Modulo de Infraestructura Interno de un agente en RETSINA**
MAS Interoperation RETSINA-OAA Interoperator	
Capability to Agent Mapping Matchmaker	Capability to Agent Mapping Matchmaker Module
Name to Location Mapping ANS	Name to Location Mapping ANS Module
Security Certificate of Authority/Cryptography Services	Security Security Module/ private/public keys
Performances Services Failure Monitoring	Performances Services Self-Monitory/Cloning
MAS Management Services Logger/ActivityVisualizer/Launcher	Management Services Logger Module
ACL Infrastructure Public Ontology/Protocols Servers	ACL Infrastructure ACL Parser/Private Ontology/Protocol Engine
Communication Infrastructure Discovery/Message Transfer	Communication Modules Discovery Module/RETSINA Communicator
Operating Environment Machines, OS, Network Multicast Transport Layer: TCP/IP, Wireless, Infrared, SSL	

Tabla 2.2: Niveles de la arquitectura RETSINA

La activación dinámica de los agentes forma una estructura organizacional creada bajo demanda que se adecua a las tareas, las necesidades de información del usuario y la información descompuesta requerida por los agentes de software relacionados. La organización de los agentes no cambia con la incorporación o eliminación de las fuentes de información, pero la interacción entre los agentes podría verse afectada por la incorporación o eliminación de los agentes.

La organización se caracteriza por la existencia de un número finito de agentes de tareas con los cuales cada agente se comunica. Además, los agentes de tareas son eventualmente responsables de resolver conflictos con la información e integrar información de fuentes heterogéneas para sus respectivas tareas. Estos agentes son

responsables de activar los agentes de información relevantes y de la coordinación de la búsqueda y filtrado de información para sus tareas. Cuando un agente de tarea recibe una tarea desde un agente de interfaz u otro agente de tarea, este la descompone teniendo en cuesta su conocimiento del dominio y posteriormente delega las subtareas a otros agentes de tarea o directamente a agentes de información.

Cuando las fuentes de información están parcialmente replicadas con distintos grados de confiabilidad, costo y tiempo de procesamiento, los agentes de información deben optimizar la selección de la fuente de información. Si las fuentes de información seleccionadas fallan en proporcionar una respuesta útil, los agentes de información deberían buscar y tratar con otras fuentes para repetir la consulta de datos. Debido a estas complejidades es que se ve a la recuperación de información como una tarea de planificación. Este tipo de agentes inteligentes difiere de los sistemas tradicionales de IA en que la búsqueda de la información durante la resolución del problema es una parte inherente del sistema. Esto se debe a que la información recuperada puede cambiar la forma en que ve el mundo externo el planificador o el sistema de confianza interno del planificador.

En RETSINA la comunicación se lleva a cabo utilizando ejecutivas KQML. En el caso de los sistemas heredados se construye un agente gateway que realiza la conexión entre dicho sistema y los agentes de RETSINA.

Los agentes están siempre localizados. El mecanismo de localización se basa en el concepto de las páginas amarillas, o sea, un agente que contiene información que se actualiza dinámicamente sobre que agentes hay en la organización. Este agente se denomina matchmaker y ante el se presentan los agentes que se van incorporando a la organización. Un agente que requiera los servicios de otro agente le solicita al matchmaker una lista con los agentes que proporcionan ese servicio.

Un matchmaker es un agente de información que puede localizar a otro agente en lugar de piezas de información. Si se da el caso que matchmaker no conoce a un agente que proporcione un determinado servicio requerido, el agente que hizo el requerimiento puede encargarle al matchmaker una monitorización para que este le avise cuando se encuentre disponible un agente que preste ese servicio. Esta es una propiedad que se deriva de que el matchmaker es una agente de información.

2.6 Conclusiones

En este capítulo se han presentado diversas arquitecturas que permiten describir el comportamiento de un agente. Básicamente existen dos perspectivas extremas, la deliberativa y la reactiva, y múltiples intermedias (todas aquellas que pertenecen a la categoría de híbridas). Los dos tipos de arquitecturas reflejan dos ideas básicas del concepto de agente: la necesidad de reflexión, para un razonamiento a largo plazo, y la urgencia de la reacción, para un comportamiento adecuado a la situación actual del agente. Se han presentado los fundamentos de los dos tipos de arquitecturas. Se ha hecho especial hincapié en la arquitectura deliberativa basada en deseos, creencias e intenciones (BDI) y la arquitectura reactiva se ha presentado desde un punto de vista más centrado en las aplicaciones robóticas, por ser este el campo en el que se desarrolló y donde el número de sistemas construidos con ella es mayor. Por otra parte se han revisado los aspectos más importantes a tener en cuenta en el desarrollo de los sistemas multiagente y se han analizado sus limitaciones. También se ha presentado la propuesta realizada por FIPA, que actualmente se puede considerar el estándar más ampliamente difundido tanto en lo que se refiere a interoperabilidad para las comunicaciones entre

agentes, como para la realización de plataformas y servicios para la ejecución y despliegue de agentes. Finalmente se ha presentado RETSINA, un ejemplo de arquitectura multiagente, estudiado y representativo de este tipo de tecnología.

Referencias

Berlanga A., A. Sanchis, P. Isasi, J.M. Molina. "Generalization Capabilities of Co-evolution in Learning Robot Behavior", Journal of Robotic Systems. vol 19, nº 10 pp 455 - 467, Octubre 2002.

Berlanga A., A. Sanchis, P. Isasi, J.M. Molina. "Neural Network Controller against Environment: A Coevolutive approach to Generalize Robot Navigation Behavior", Journal of Intelligent and Robotic Systems. vol 33, nº2 pp 139-166, 2002.

Berlanga A., Borrajo D., Fernández F., García R., Molina J. M., Sanchis A.. (1999) "Robótica Cognoscitiva y Aprendizaje Automático", VIII Conferencia de la Asociación Española para la Inteligencia Artificial y III Jornadas de Transferencia Tecnológica de Inteligencia Artificial. SE sobre Agentes Autónomos. pp. 1-8. Murcia, España. Noviembre 1999.

Bonasso A., Peter R., Kortenkamp D. y Miller D. (1995) 'Experiences with An Architecture for Intelligent, Reactive Agents', INTELLIGENT AGENTS II: Agent Theories, Architectures and Languages, pp 187-203, Wooldridge, Michael J, Mueller, P and Tambe, Milind, (Eds), Springer Verlag, Berlin 1995.

Bond A.H., Gasser L. (1988) Readings in Distributed Artificial Intelligence. Morgan Kaufman.

Braitenberg, V., (1984) Vehicles. Experiments in Synthetic Psycology. The MIT Press, Cambridge, Massachusetts, 1984.

Bratman M. (1990) ´What is intention? In Intentions in Communication, MIT Press, Cambridge, pp 15-31, Cohen P.R., Morgan J., and Pollack M.E. (Eds). 1990.

Brooks R. A. (1990) 'Elephants Don't Play Chess', DESIGNING AUTONOMOUS AGENTS: Theory and Practice From Biology to Engineering and Back, pp 3-17, Maes, Pattie, (Ed), Cambridge, MA, The MIT Press, 1990.

Brooks R. A. (1991) "Intelligence without Representation". Artificial Intelligence, 47, 139-159.

Brooks R. A. (1991), 'Intelligence Without Reason', PROCEEDINGS OF IJCAI, 1991, Sydney, Australia.

Brooks, R.A., (1986) A Roboust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation RA-2, 14-23, 1986.

Camacho D., D. Borrajo, J. M. Molina. (2001) "Intelligent Travel Planning: A Multiagent Planning System to Solve Web Problems in the e-Tourism Domain", Autonomous Agents and Multi-Agent Systems. 4 (4), pp 387-392, Diciembre 2001.

Cammarata S., McArthur D., Steeb R. (1998) "Strategies of Cooperation in Distributed Problem Solving", en Readings in Distributed Artificial Intelligence, Ed. Alan H. Bond and Les Gasser, Morgan Kaufmann 1988.

Canamero D. (1997) 'Modeling Motivations and Emotions as a Basis for Intelligent Behaviour', AUTONOMOUS AGENTS '97, Proceedings of the First International Conference on Autonomous Agents, pp 148-156, Marina del Rey CA, Feb 5-8, 1997, ACM Press.

Carbó J., J. M. Molina, J. Dávila (2003) "Trust Management through Fuzzy Reputation", International Journal of Cooperative Information Systems. vol 12, nº 1 pp 135-155, Marzo 2003.

Castillo, L.F., Corchado, J.M.. and Glez-Bedia, M. Models for Planning in real-time domains. 2nd International Workshop on Practical Applications of Agents and Multi-agents Systems (IWPAAMS'2003). VALLADOLID, Spain, October, 15-17, 2003

Cavendon L. y Rao A. S. (1997) Bringing about rationality: incorporating plans into a BDI Agent Architecture.

Chellas B. F. (1980) Modal Logic: an introduction. Cambridge University Press, Cambridge, 1980.

Cohen P. R. y Levesque H. J. (1990) Intention is choice with commitment. Artificial Intelligence, 42. 1990.

Cohen P. R. y Levesque H. J. (1990) Rational interaction as the basis for communication. In Intentions in Communication, P.R. Cohen, J. Morgan and M.E. Pollack (Eds). The MIT Press. 1990.

Cohen P. R. y Levesque H. J. (1995) Communicative Actions for Artificial Agents. Center for Human-Computer Communication. Dept. of Computer Science and Engineering. Oregon Graduate Institute of Science and Technology.

Cohen, P.R., Greenberg, M.L. Hart, D.M. and Howe, A.E. (1989). Trial by FIRE: Understanding the design requirements for agents in complex environments. AI Magazine, 10(3), pp.32-48, 1989

Conry S., Meyer R.A., Lesser V., (1988) "Multistage Negotiation in Distributed Artificial Intelligence". En *Readings in Distributed Artificial Intelligence*, Editores Bond y Gasser, Morgan Kaufman, 1988.

Conte R., Miceli M., Castelfranchi C. (1991) "Limits and Levels of Cooperation: Disentangling Various Types of Prosocial Interaction", *Distributed AI 2*. Editado por Demazeau Y., Muller J.P., North Holland, 1991.

Corchado J. M. and Laza R. (2003) Constructing Deliberative Agents with Case-based Reasoning Technology. International Journal of Inteligent Systems. Vol 18, No. 12, December. ISSN 0884-8173.

Corchado J. M. And Laza R. (2003). Constructing Deliberative Agents with Case-based Reasoning Technology, International Journal of Intelligent Systems. Vol 18, No. 12, December.

Corchado J. M., Pavón P., Corchado E. S. and Castillo L. F. (2004) Development of CBR-BDI Agents: A Tourist Guide Application. European Conference on Case-based Reasoning 2004. Springer Verlag.

Davis, R., Smith, R.G., "Negotiantion as a Metaphor for Distributed Problem Solving", Artificial Intelligence vol 20, nº 1, pp 63-109, 1983.

Demazeau Y., Muller J.P., (1991), *Idecentralized AI 2*, Elsevier North Holland.

Dennet D. C. (1987) The intentional Stance. The MIT Press, 1987.

Durfee (1988) Coordination of Distributed Problem Solvers. Boston, Kluwer Academic.

Durfee E.H., Lesser V. R. (1991) "Global Partial Planning: A Coordination Framework for Distributed Hypothesis". IEEE Transactions on Systems, Man and Cybernetics. 21(6).

Durfee E.H., Lesser V.R., Corkill D.D., (1989) "Cooperative Distributed Problem Solving", Cap. XVII, The Handbook of AI (Vol IV), ed. Barr A., Cohen P.R., Feigembaum E.A., Addison-Wesley 1989

Ferber, J., (1999) *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*, Addison-Wesley.

Ferguson I. A. (1992) TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents. PhD thesis, clare Hall, University of Cambridge, UK, November 1992.

FIPA. Foundation for Intelligent Physical Agents. http://www.fipa.org.

Gasser L., Braganza C., Herman N., "MACE: a Flexible Testbed for Distributed AI Research", *Distributed Artificial Intelligence*, Editado por M.N. Huns, pp 119-152, Pitman (1987).

Gasser L., Hill R. W. (1990) "Coordinated Problem Solvers". Annual Reviews of Computer Science. 4, pp 203-253.

Georgeff M. P. (1983) "Communication and Interaction in Multi-Agent Planning". Proc. of AAAI-83, pp 125-129.

Georgeff M. y Lansky A. L. (1987) Reactive reasoning and planning. Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87). Seattle, WA, 1987.

Georgeff M., Pollack M. y Tambe M. (1999) The Belief-Desire-Intention Model of Agency. Proceedings of the Fifth International Workshop on Agent Thories, Architectures and Languages (ATAL-98), Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg, 1999.

Georgeff. M. y Rao A. (1998) Rational Software Agents: From Theory to Practice. In Agent Technology: Foundations, applications and markets. Jennings & Wooldridge (Eds). Pag. 139-160. Springer, 1998.

Giampapa J.A., Sycara K., Conversational Case-Based Planning for Agent Team Coordination, Robotics Institute, Carnegie Mellon University.

Glez-Bedia M, Corchado J. M., Corchado E. S. and Fyfe C. (2002). Analytical Model for Constructing Deliberative Agents, Engineering Intelligent Systems, Vol 3: 173-185.

Glez-Bedia M. y Corchado J. M. (2003) Constructing autonomous distributed systems using CBR-BDI agents. Innovation in knowledge Engineering. Faucher C., Jain L. and Ichalkaranje N. (eds.).

Goodridge, S. G., Luo, R.C. (1994) Fuzzy Behavior Fusion for Reactive Control of an Autonomous Mobile Robot: MARGE Proccedings of IEEE Int. Conference on Robotics and Automation, 1622-1627, 1994.

Grosz B. y Sidner C. (1990) Plans for discourse. Intentions for Communication, pp 417-444, MIT Press.

Haddadi A. y Sundermeyer K. (1996) ´Belief-Desire-Intention Agent Architectures´, Foundations of Distributed Artificial Intelligence, pp 169-185, O´Hare G. M. P.; Jennings N. R. (Eds) Wiley-Interscience Publication. 1996.

Hayes-Roth, B. (1988). Making Intelligence systems adaptive. Report STAN-CS-88-1226, Standford University, Standford, CA, October, 1988; Hayes-Roth, B. (1990). Architectural foundations for real-time performance in intelligent agents. The Journal of Real-Time Systems, 2, pp.99-125, 1990.

Hoare C. A. R. (1985) Communicating sequential processes. Prentice Hall International.

Hughes G. E. y Cresswell M. J. (1996) A new introduction to Modal Logic. Routledge, London, 1996

Jennings N. R. (1993) Specification and implementation of a belief-desire-joint-intention architecture for collaborative problem solving. Int, J. Intell. Coop. Inf. Syst., 289-318. 1993.

Keith D., Pannu A., Sycara K. y Williamson M. (1997) 'Designing Behaviors for Information Agents', AUTONOMOUS AGENTS '97, Proceedings of the First International Conference on Autonomous Agents, pp 404-413, Marina del Rey CA, Feb 5-8, 1997, ACM Press.

Kinny D., Georgeff. M. (1993) Commitment and effectiveness of situated agents. Proceedings of Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93) Chambery, France, 1993.

Kinny D., Rao A. S, Georgeff, M. P. (1996) A methodology and modelling technique for Systems of BDI agents. Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW96). Lecture Notes in Artificial Intelligence 1038. 1996.

Kolodner, JL. (1983). Case-Based Reasoning Workshop, Morgan Kaufmann, Calif., US

Lesser V. R. (1991) "A Retrospective View of FA/C Distributed Problem Solving". IEEE Transactions on Systems, Man and Cybernetics. 21 (6) 1347-1363.

Maes P. (1989) 'Situated Agents Can Have Goals', Designing Autonomous Agents: Theory and Practice From Biology to Engineering and Back, pp 49-71, Maes, Pattie, (Ed), 1989

Malone T. W. (1988) "What is coordination theory?". National Science Foundation Coordination Theory Workshop, MIT.

Mataric M. J. (1992) "Designing Emergent Behaviors: from local Interaction to Collective Intelligence". From Animals to Animats 2, Proc. of the Second International Conference Simulation of Adaptive Behavior. Hawaii. Editores J-A Meyer, H.L. Roitblat y S.W. Wilson pp. 432-441, MIT Press.

Mataric M.J., (1994) "Learning to Behave Socially", From Animals to Animat 3, Proc. Of the third International Conference on Simulation of Adaptive Behavior, Brighton, Cliff, Husband, Meyer y Wilson Ed., pp 453-462, MIT Press.

Matellán V., C. Fernández and J. M. Molina. (1998) "Genetic Learning of Fuzzy Reactive Controllers", In Robotics and Autonomous Systems, vol 25, nº 1-2, pp 33-41, Noviembre 1998.

Matellán V., Molina J. M. and Fernández C. (1995) “Fusion of Fuzzy Behaviours for Autonomous Robots”. 3rd International Symposium on Intelligent Robotic Systems, pp 157-163. Pisa, Italia. Julio 1995.

Matellán V., Molina J. M. and Sommauga L. (1996), “Fuzzy Cooperation of Autonomous Robots”. 4th International Symposium on Intelligent Robotic Systems, pp 377-384. Lisba, Portugal. Julio 1996.

Matellán V., Molina J. M., Fernández C. and Blázquez J. M. (1995). “Integración de Controladores Difusos en un Robot Autónomo”, V Congreso Español sobre Tecnología y Lógica Fuzzy, pp 221-226. Murcia, España. España. Septiembre 1995.

Molina J. M. and Matellán V. (1996). “Robots Autónomos: Arquitecturas y Control”, BURÁN Nº 7, pp 19-24. ABRIL 1996. Editada por la Rama de Estudiantes del IEEE, 1996.

Molina J. M., Galván I.M., Valls J.M. and Leal A.. “Optimizing the Number of Learning Cycles in the Design of Radial Basis Neural Networks using a Multi-Agent System”, Computing and Informatics. vol 20, nº5 pp 429-449, 2001.

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

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

Molina J. M., Sanchis A., Berlanga A. and Isasi P.. “An Enhanced Classifier System for Autonomous Robot Navigation in Dynamic Environments”, Intelligent Automation and Soft Computing. Autosoft Press, vol 6, nº 2, pp 113-124, 2000.

Moore R. C. (1990) A formal Theory of knowledge and action. In J.F. Allen, J. Hendler and A. Tate, editors. Readings in Planning. Morgan Kaufmann, 1990.

Müller J. P. (1997) The Design of Intelligence Agents (LNAI 1037). Springer-Verlag: Berlin, Germany, 1997.

Ogasawara G.H. (1991). A distributed, decision-theoretic control system for a mobile robot. ACM SIGART Bulletin, 2(4), pp. 140-145, 1991.

Parunak H. V. D. (1987) “Manufacturing Experience with the Contract Net”. Distributed Aritificial Intelligence vol 1, Editor M. Huhns, pp. 285-310, Londres, Pitman.

Payne, T.R., Lenox T.L., Hahn S., Sycara K. and Lewis M. (????) Agent-Based Support for Human/Agent Teams, Robotics Institute, Carnegie Mellon Univesity, School of Information Sciences, Universtiy of Pittsburg.

Rao A. and Georgeff M., (1992) “Social Plans: Preliminary Report”, Proc. Of MAAMAW’91, Werner, Castelfranchi (Ed.), pp 127-146, Elsevier North Holland.

Rao A. S. and Georgeff M. P. (1991) Asymmetry thesis and side-effects problems in linear time and branching time intention logics. Proceedings of the twelfth International Joint Conference on Artificial Intelligence (IJCAI-91) Sidney, Australia. 1991.

Rao A. S. and Georgeff M. P. (1991) Modeling Rational Agents within a BDI-Architecture. Second International Conference on Principles of Knowledge Representation and Reasoning (KR91). San Mateo, C.A. 1991.

Rao A. S. and Georgeff M. P. (1992) An abstract architecture for rational agents. In Proceedings of Third International Conference on Principles of Knowledge Representation and Reasoning .Morgan Kaufmann Publishers, San Mateo, C.A. 1992.

Rao A. S. y Georgeff M. P. (1995) BDI Agents from Theory to Practice. Proceedings of the First International Conference on Multi-Agents Systems. (ICMAS-95). San Francisco. June 1995.

Rao A. S. y Georgeff M. P. (1998) Decision procedures of BDI. Logics. Journal of logic and computation 8(3). 1998.

Reynolds C. (1987) “Flocks, Herds and Schools: A Distributed Behavioral Model”. Computer Graphics. 21 (4) pp. 25-34.

Rosenschein S. and Kaelbling, L. P. (1986) The synthesis of digital machines with provable epistemic properties. In J.Y.Halpen, editor. Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge. Morgan Kaufmann, 1986.

Salido, M and Barber, F. (2002) A Polynomial Algorithm for Continuous Non-binary Disjunctive CSPs.

Sandholm T., Lesser V. R. (1995) “Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Protocol”. Proc. of the Second International Conference on Multiagent Systems, 328-335. Menol Park, California, AAAI Press.

Schild K. (1999) On the relation between standard BDI logics and standard logics of concurrence. Proceedings of the Fifth International Workshop on Agent Thories, Architectures and Languages (ATAL-98), Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg, 1999.

Seel N. (1989) Agents Theories and architectures. PhD. Thesis. Surrey University, Guildford, UK. 1989.

Singh M.P., (1994) “Multiagent Systems: A Theoretical Framework for Intentions, Know How and Communications”, Lecture Notes in Artificial Intelligence 799, Springer Verlag.

Smith R.G., (1979) “A Framework for Distributed Problem-solving”, Proc. Of IJCAI ’79.

Smith R.G., (1980) “The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver”, IEEE Trans. On Computers, vol 29, nº12, pp 1104-1113.

Sommaruga L., Merino I., Matellán V. and Molina J. M. (1996). “A Distributed Simulator for Intelligent Autonomous Robots”. 4th International Symposium on Intelligent Robotic Systems, pp 393-399. Lisba, Portugal. Julio 1996.

Steel L., (1989) “Cooperation between Distributed Agents through Self-Organization”, Decentralized AI, Demazeau, Muller (ed.) Elsevier.

Steel L., (1994) “A Case Study in the Behavior-Oriented Design of Autonomous Agents”, *Proc. Of the ’94 Conference on Simulation of Adaptive Behavior*, MIT Pres.

Suthankar G. and Sycara K. (????) Team-aware Robotic Demining Agents for Military Simulation, Robotics Institute, Carnegie Mellon University.

Sycara K. (1989) “Multiagent Compromise via Negotiation”. En *Distributed Artificial Intelligence*, Editores Gasser y Huns, Pitman, 1989.

Sycara K. (1997) “Using Option Pricing to Value Commitment Flexibility in Multiagent Systems. Technical Report CMU-CS-TR-97-169, School of Computer Science, Carnegie Mellon University.

Sycara K., Decker, K., Pannu A., Williamson M., and Zeng D. (1996) “Distributed Intelligent Agents." IEEE Expert: Intelligent Systems and Their Applications. Vol. 11, No. 6, Dec., 1996, pp. 36-46.

Sycara K., Paoluci M., van Velson M. and Giampapa J. (????) The RETSINA MAS Infraestructure, Robotics Institute, Carnegie Mellon University.

The Intelligent Software Agents Group, The Robotics Institute, Carnegie Mellon University, RETSINA demining, 2001, http://www2-cs.cmu.edu/~softagents/demining.html

Von Martial F. (1992) Coordinating Plants of Autonomous Agents. Lecture Notes in Artificial Intelligence No. 610. Springer-Verlag, Berlin.

Wavish P. R. and Connah D. M. (1990) “Representing Multi-Agent Worlds in ABLE”. Technical Note, TN2964, Philips Research Laboratories.

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.

Wooldridge M. and Jennings N. R. (1994) ‘Agent Theories, Architectures and Languages: A Survey’, Intelligent Agents (ECAI-94 Workshop Proceedings on Agent Theories, Architectures and Languages, Amsterdam, Aug, 1994), pp 1-22, Wooldridge Michael J, and Jennings, Nicholas, (Eds), Springer Verlag, Berlin.

Wooldridge M. and Jennings N. R. (1995) Intelligent Agents: Theory and Practice. The Knowledge Engineering Review, vol. 10(2) pp. 115-152, 1995.

Wooldridge, M. (1992) The logical modelling of computational Multi-Agents Systems. PhD. Thesis, Department of Computation, UMIST, Manchester, UK. October 1992.

Zlotkin G. and Rosenschein J. S. (1993) One, two, many coalitions in multi-agent systems. Proceedings of the Fifth European Workshop on Modelling Autonomous Agentes in a Multi-Agent world (Ghedira and K. Sprumont, eds.),U. of Neuchatel, Switzerland.