Arquitecturas y Comunicación entre Agentes

Ramón Rizo, Faraón Llorens y Mar Pujol
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
{rizo,faraon,mar}@dccia.ua.es

1 Sistemas Multiagente

Los *Sistemas MultiAgente* (MAS "MultiAgent System") forman una comunidad social de miembros interdependientes que actúan individualmente. Desde el momento en que un grupo de seres, en este caso agentes, asumen vivir en grupo formando sociedades, es necesaria la capacidad de negociar y coordinar diferentes tareas.

En muchas ocasiones el uso de agentes individuales no es igual de adecuado para todas las situaciones que ocurren en la práctica. La resolución de un problema utilizando un agente individual causa grandes restricciones. Un agente individual requiere una enorme cantidad de conocimiento para resolver problemas complejos. En el peor de los casos, el problema puede ser tan complejo que un agente no pueda encontrar una solución útil. Incluso cuando el agente individual es capaz de resolver un problema, siempre presenta un cuello de botella en lo que se refiere a velocidad, fiabilidad, flexibilidad y modularidad.

Los sistemas multiagente ofrecen un método para evitar las situaciones problemáticas descritas. En un sistema multiagente están activos diversos agentes autónomos independientes [Brenner,1998]. Cada uno de estos agentes se dedica a sus propios objetivos y sólo contacta con los otros agentes para obtener información, o para contribuir a una solución coordinada de un problema general. En ambas situaciones, cada agente individual tiene una tarea específica para la cual es adecuado y cuya solución no excede de sus capacidades. Esto permite el procesamiento de problemas complejos.

Los sistemas multiagente proporcionan una gran ventaja: permiten la integración de agentes existentes en un gran sistema. Por tanto, la solución de un problema no requiere el diseño y desarrollo de un nuevo agente especializado, en su lugar, puede ser utilizado el conocimiento de los agentes existentes combinándolos en un sistema multiagente y permitiendo que trabajen conjuntamente para resolver el problema.

1.1 Interacción entre agentes

En un sistema multiagente un agente debe prever las acciones de los otros agentes en su propia tarea de planificación y cómo puede influir en las acciones de otros agentes en beneficio de sus propios objetivos. Los efectos de las acciones de otros agentes sobre nuestro agente pueden ser favorables, neutros o perjudiciales para los objetivos de nuestro agente. Para influir sobre lo que otro agente hará necesitaremos métodos para que un agente se pueda *comunicar* con otro agente.

La clase más sencilla de interacción entre agentes sería aquella en la que nuestro agente reacciona a los efectos del entorno generados por los demás agentes a medida que ocurriesen y los percibiera. Se trata de *agentes reactivos*. En este caso no existiría comunicación explícita entre ellos, ni necesitaríamos modelos de los otros agentes. Si nuestro agente ha de tener en cuenta la actividad de los otros agentes, necesitará disponer de modelos que pueda usar para prever cómo se comportarán estos y utilizarlos al construir sus propios planes. Al modelo del agente, junto con el mecanismo para utilizarlo y seleccionar las acciones, se le denomina *estructura cognitiva*. A menudo, la estructura cognitiva incluye también los objetivos e intenciones del agente [Nilsson,2001]. Todo esto lo trataremos con más detalle en el apartado 3 al hablar del las arquitecturas de agentes.

Algunas características básicas de los sistemas multiagente son:

- Cada agente tiene una vista limitada del estado del mundo, es decir, tiene información incompleta.
- No hay un control global, el control del sistema está distribuido.
- La información está descentralizada.
- La computación es asíncrona.

En un sistema multiagente podemos encontrarnos en situaciones de *cooperación* o de *competición*. En el primer caso, los agentes van a tener que ser capaces de combinar sus esfuerzos, normalmente para obtener soluciones de problemas que el propio agente, por sí mismo, no es capaz de encontrar (CMAS "Cooperative Multiagent Systems"). Por otro lado, desde el momento en que nosotros tenemos agentes que cumplen nuestras órdenes, otras personas tendrán agentes que cumplirán las suyas. Y mientras nuestros agentes quieren lo mejor para nosotros, los de los demás también intentarán lo mejor para sus propietarios (SMAS "Self-Interested Multiagent Systems"). En este último tipo de sistemas juegan un papel preponderante las estrategias de negociación que permiten resolver conflictos, asignar tareas y tomar decisiones (este aspecto será tratado en el apartado 2.7).

Los protocolos de interacción entre agentes (AIP, “Agent Interaction Protocols”) suelen ser bastante complejos. Un protocolo de interacción entre agentes describe un patrón de comunicación como una secuencia permitida de mensajes entre agentes y las

restricciones en el contenido de dichos mensajes. Para ayudarnos podemos representarlos mediante AUML (Agent UML - "Agent Unified Modeling Language") [Odell2001], [Url_AUML]. Se trata de una extensión de UML para tratar con los agentes. Se estructura en tres capas y extiende las siguientes representaciones:

- *Plantillas* ("templates") y *paquetes* ("packages"): representan el protocolo completo
- *Diagramas de secuencia* ("sequence diagrams") y *colaborativos* ("collaborative diagrams"): capturan la dinámica entre agentes (inter-agent)
- *Diagramas de actividad* ("activity diagrams") y *gráficos de estado* ("statecharts"): capturan las dinámicas intra e inter agentes

1.2 Inteligencia Artificial Distribuida y Resolución de Problemas

Muchos problemas tienen una naturaleza distribuida, por esta razón requieren de una solución distribuida. El conocimiento especializado en muchas ocasiones no esta disponible en un agente individual pero puede estar distribuido entre diferentes agentes [Llorens,2001]. Como ya hemos dicho en el apartado anterior, en la interacción entre agentes puede darse el caso de que diversos agentes coordinen sus tareas para alcanzar un objetivo común. Este aspecto es tratado en la *Inteligencia Artificial Distribuida* (IAD), que la podríamos definir como [Weiss,2000]:

La inteligencia artificial distribuida es el estudio, la construcción y aplicación de los sistemas multiagente, esto es, sistemas en los cuales varios agentes inteligentes, interactuando, persiguen un conjunto de objetivos o realizan un conjunto de tareas.

Tradicionalmente se distinguen dos tipos básicos de sistemas: los sistemas multiagente y los sistemas de resolución distribuida de problemas. Mientras que en los sistemas multiagente el énfasis se pone en la coordinación, en los sistemas de resolución distribuida de problemas está en la tarea de descomposición y de síntesis de las soluciones. En la actualidad no existe tal diferenciación y al hablar de sistemas multiagente englobamos ambos tipos.

La solución de problemas distribuidos con sistemas multiagente es apropiada sólo en el caso en que los agentes integrados en el sistema sean capaces de comunicarse y cooperar con los otros.

Así, la metodología usada para resolver un problema global de forma distribuida, consta de tres pasos secuenciales: división del problema en subproblemas, solución de los subproblemas y combinación de las subsoluciones. Veámoslos con más detalle:

1. La división del problema global en una serie de problemas parciales que deberán asignarse posteriormente a determinados agentes es el paso más importante. Por una parte, se debe decidir cómo dividir de forma concreta el problema y de qué manera. Esta tarea puede ser realizada por un agente especializado para la tarea, o ser realizada entre todos los agentes. Posteriormente, hay que encontrar a agentes capaces de resolver cada subproblema. Si éstos se asignan a agentes individuales, no existe ningún inconveniente; pero si hay varios agentes capaces de resolver el mismo problema (esto es, de conocimientos similares), serán necesarias estrategias de cooperación entre ellos o toma de decisiones en la asignación de subproblemas.
2. La resolución de los subproblemas. Como cada agente es responsable de la resolución del problema asignado, en principio no es necesaria ninguna comunicación ni cooperación entre ellos. Sin embargo, a menudo esto no sucede, puesto que normalmente el agente no va a ser capaz de resolver el problema de forma autónoma. Los agentes deberán preguntar a otros por problemas que ellos mismos no serán capaces de resolver. A veces, el agente sabrá a qué agente (o conjunto de ellos) debe preguntar; otras veces, el agente no sabrá a quién preguntar, con lo que necesitará del concurso de otros agentes que buscarán por él agentes que posean el conocimiento requerido.
3. La síntesis de las soluciones parciales para generar una solución al problema global. Este paso genera problemas parecidos al primer paso; esto es, será necesario un agente capaz de sopesar la importancia de las subsoluciones y mezclarlas para obtener una solución concreta. Esto no siempre tiene por qué suceder; a veces, la combinación de subsoluciones no generará una solución completa.

Si cada agente individual es responsable de la solución de su subproblema, el segundo paso del proceso de solución no requiere necesariamente de procesos de comunicación y cooperación. Pero en muchas ocasiones la interacción entre diferentes agentes es necesaria porque los subproblemas individuales no pueden resolverse de forma autónoma. Por esta razón, pueden aparecer conflictos directos entre dos o más agentes, lo que requiere que exista comunicación entre los agentes para resolver el conflicto. Así mismo, los agentes deben comunicarse entre sí para alcanzar la solución general del problema.

2 Comunicación entre Agentes

El problema que surge cuando dos seres inteligentes intentan interactuar es que necesitan unas normas y un canal de comunicación: deben de utilizar el mismo lenguaje, estar de acuerdo en el significado de los símbolos de ese lenguaje, tener un mecanismo

de comunicación para el intercambio de mensajes, no hablar al mismo tiempo, etc. En resumen, debe de existir comunicación y coordinación entre ellos.

2.1 Actos de Habla

Nuestro agente se puede comunicar con otros agentes, de forma que pueda provocarles cambios en sus objetivos y/o creencias. Decimos que dos agentes han establecido un *acto comunicativo* cuando un agente lleva a cabo una serie de acciones que tienen como objetivo modificar la estructura cognitiva del otro agente.

Debido a que la mayoría de los actos comunicativos entre personas se realizan mediante el lenguaje hablado, los expertos en lingüística utilizan el término *acto de habla* para referirse a todos los tipos de actos comunicativos [Searle,1969], [Searle,2001]. Por la misma razón, al agente que toma la iniciativa de la comunicación lo llamaremos *hablante* y al agente al que va dirigida dicha acción, *oyente*. Existen varias categorías de actos de habla:

- *Representativos*: exponen una determinada proposición como representación de un estado de cosas del mundo.
- *Directivos*: mediante los que se realiza una petición, se da una orden o se hace un ruego. En ellos se intenta que el oyente actúe de tal modo que su conducta concuerde con el contenido proposicional.
- *Exhortativos*: para prometer, comprometer, jurar, amenazar o realizar contratos o garantías. Constituye un compromiso por parte del hablante de adoptar el tipo de acción representado en el contenido proposicional.
- *Expresivos*: para expresar un agradecimiento, felicitaciones, bienvenidas o pedir disculpas.
- *Declaraciones*: actos que producen un cambio en el estado del mundo.

La teoría de actos de habla ha contribuido en gran medida al entendimiento de la relación entre el estado interno de un agente y las expresiones que intercambia con otros agentes. La teoría se basa en la observación de que las oraciones expresadas por humanos durante la comunicación no únicamente aseveran un hecho, sino que en realidad tratan de transmitir una creencia o conocimiento, una intención o un deseo. En general, los actos de habla provocan ciertos efectos sobre el conocimiento del oyente. Una *sentencia* es la manifestación de un acto verbal y debe reflejar el contenido proposicional y el tipo de acto de habla que manifiesta. Se supone que los actos de habla provocan ciertos efectos sobre el conocimiento del oyente. El efecto que el hablante intenta provocar en el oyente se denomina *efecto perlocutivo* del acto de habla y el efecto que realmente produce, *efecto ilocutivo*.

Un acto comunicativo debe trasmitirse desde el hablante al oyente en forma de sentencia. Para ello utilizaremos un *lenguaje de comunicación* común, preestablecido de mutuo acuerdo. Pero lo realmente importante en el acto comunicativo para el intercambio de conocimiento consiste en integrar una semántica en el lenguaje de comunicación. Para ello, se crean lenguajes basados en la teoría de actos de habla, con la que se construye una capa lingüística y se formalizan las acciones lingüísticas de los agentes.

2.2 Coordinación entre Agentes

Cuando el número de agentes del sistema crece, aparece la necesidad de ayuda para localizar otros agentes que tengan la información o que nos puedan ofrecer los servicios que requerimos [Omicini,2001]. Muchos sistemas multiagente utilizan *agentes mediadores* o *facilitadores* que proporcionan determinados servicios y asistencia a los agentes del sistema (matchmaker, broker, blackboard, etc.). En general podemos distinguir tres categorías de agentes: P-agentes o *agentes proveedores*, R-agentes o *agentes requeridores* y *agentes mediadores* o *intermediarios*.

La comunicación forma la base de la coordinación y está formada por los protocolos de comunicación y el resultado de los métodos de comunicación.

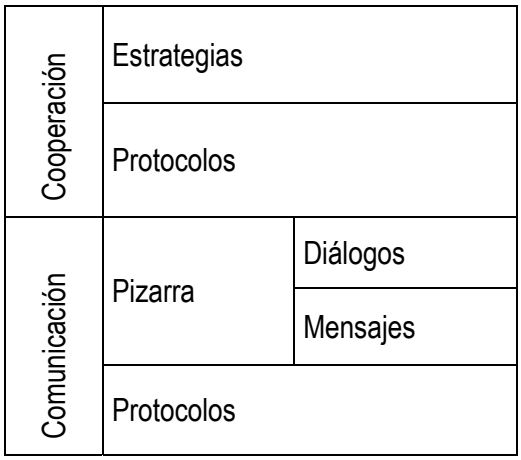

Ilustración 1: Comunicación y cooperación en sistemas multiagente

2.3 Métodos de Comunicación

Existen diferentes métodos de comunicación. La invocación del procedimiento de un agente por otro agente representa el caso más simple. No obstante, sólo métodos de comunicación simples pueden implementarse utilizando llamadas a procedimientos. Por esta razón, las llamadas a procedimientos no se consideran un método de comunicación.

Los métodos de comunicación se pueden diferenciar en sistemas de *pizarra* y sistemas de *mensaje/dialogo*. Veamos cada uno de ellos.

2.3.1 Sistemas de Pizarra

La arquitectura de pizarra es ampliamente utilizada en Inteligencia Artificial. En un sistema con múltiples fuentes de conocimiento (o agentes independientes) necesitamos un mecanismo de comunicación. La *pizarra* ("blackboard") es una estructura de datos que es usada como mecanismo general de comunicación entre las múltiples fuentes de conocimiento y es gestionada y arbitrada por un controlador [Hayes,1985], [Nii,1986a], [Nii,1986b] y [Jagannathan,1989]. Así, será un área de trabajo común a los agentes donde poder intercambiar información, datos y conocimiento. Al igual que las pizarras de nuestras aulas, son dispositivos para compartir información, con múltiples escritores y múltiples lectores.

Como cada agente trabaja con su parte del problema, acudirá a la pizarra para ver nueva información puesta por otros agentes y, a su vez, pondrá sus resultados. La forma de funcionar será: un agente inicia una comunicación escribiendo algún tipo de información en la pizarra; en ese momento, estos datos están disponibles para todos los agentes del sistema; cada agente accederá a la pizarra eventualmente para comprobar si hay información nueva; normalmente cada agente no recogerá toda la información que se vaya escribiendo en la pizarra, sino que sólo obtendrá aquélla que le interese, tal vez por pertenecer a conocimiento afín.

Como se puede observar (Ilustración 2), no hay comunicación directa entre agentes; por esto mismo, cada agente se ve obligado a resolver de forma autónoma su subproblema, bajo su responsabilidad. En este tipo básico de pizarras no existen áreas privadas; los agentes pueden escribir la información que deseen, y pueden acceder a toda la información contenida en la pizarra. Por tanto, si hay un número grande de agentes, la información contenida en la pizarra crece exponencialmente; y por otra parte, los agentes deberán buscar en una gran cantidad de información por cada acceso que realicen a la pizarra. Para optimizar este proceso, existen métodos más complejos (ampliaciones del original) para definir regiones en la pizarra, de manera que un agente

sólo ve la región de la pizarra que tenga asignada. Un sistema multiagente puede disponer de varias pizarras.

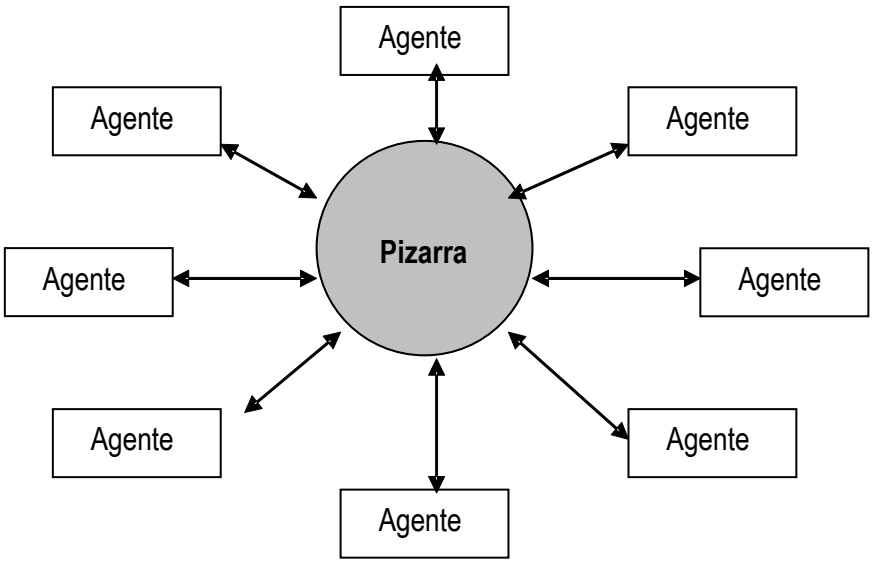

Ilustración 2: Estructura de un Sistema de Pizarra

Aunque no hay una definición específica de las estructuras de datos que debe contener la pizarra, es obvio que por lo menos se debe asegurar que el resto de agentes deben ser capaces de leer y entender los datos incluidos en el sistema. Además, se deberán incluir informaciones adicionales que faciliten el proceso de solución del problema global.

Otro dato a tener en cuenta es la calidad de las contribuciones individuales de los agentes al sistema. Como cualquier agente tiene potestad para escribir cualquier cosa en la pizarra y esto podría generar problemas, estudios posteriores de la idea original han incluido una cierta coordinación para el mantenimiento de la pizarra. Para ello, se ha introducido la idea del *moderador*, que se encarga de supervisar el estado del problema, los siguientes subproblemas a ser resueltos y además trata de organizar el trabajo entre y hacia los agentes. Cualquier agente puede usar la pizarra para leer los subproblemas generados. Si está interesado en resolver alguno, lo indica usando un registro de activación de fuente de conocimiento (KSAR, "Knowledge Source Activation Record") en una base de datos. El moderador controla y evalúa el conocimiento de la base de conocimiento para seleccionar a los agentes más capacitados para resolver el subproblema. Otra mejora es incluir un *dispatcher* que se encarga de informar a los agentes interesados de nueva información que se escribe en la pizarra, en vez de que éstos accedan regularmente a ella (Ilustración 3).

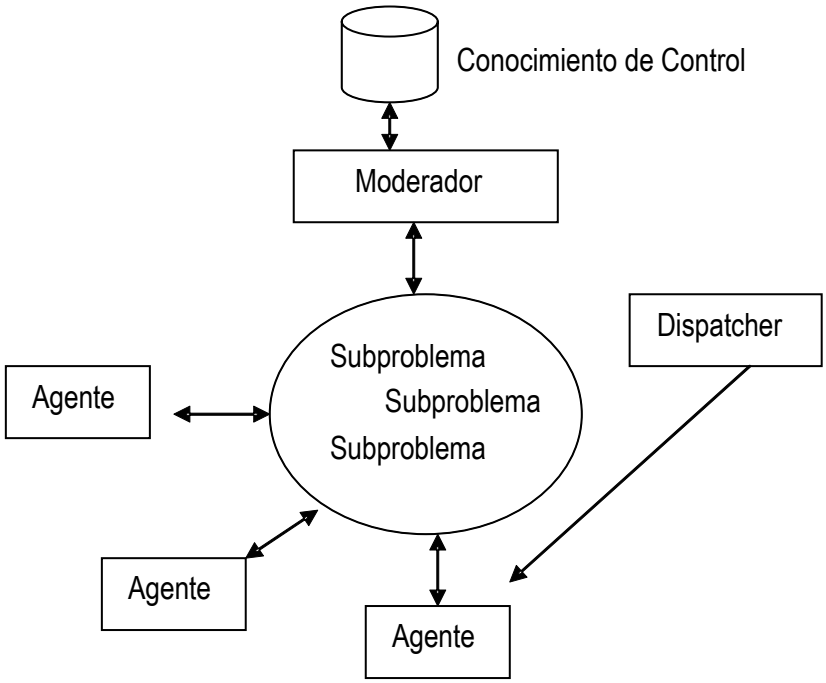

Ilustración 3: Estructura extendida de pizarra

Los sistemas de pizarra ofrecen un sistema flexible para la cooperación y comunicación entre las unidades que componen un sistema distribuido, ya que son independientes de las estrategias de cooperación. Sin embargo, la estructura central del sistema es un obstáculo para sistemas orientados a redes, puesto que la sola lectura de la información contenida en la pizarra representa un cuello de botella importante en la red.

2.3.2 Sistemas de Mensajes

La comunicación utilizando mensajes forma una base flexible para la implementación de estrategias de coordinación complejas. En los métodos basados en mensajes, los mensajes que los agentes intercambian con otros agentes pueden ser utilizados para establecer comunicaciones y mecanismos de cooperación utilizando protocolos definidos. El formato libre de los contenidos de los mensajes proporcionan capacidades de comunicación muy versátiles que no están restringidas a comandos simples.

Un agente, *emisor*, transfiere un mensaje específico a otro agente, el *receptor* (Ilustración 4). Al contrario que en los sistemas de pizarra, los mensajes son intercambiados directamente entre dos agentes. No se utiliza memoria, ni otros agentes son capaces de leer el mensaje si no va dirigido a ellos. Aunque en las situaciones normales el emisor asigna una única dirección a un mensaje, podemos utilizar el "broadcasting" como una

excepción, en la cual el mensaje es enviado a todos los agentes del sistema o a un grupo especificado.

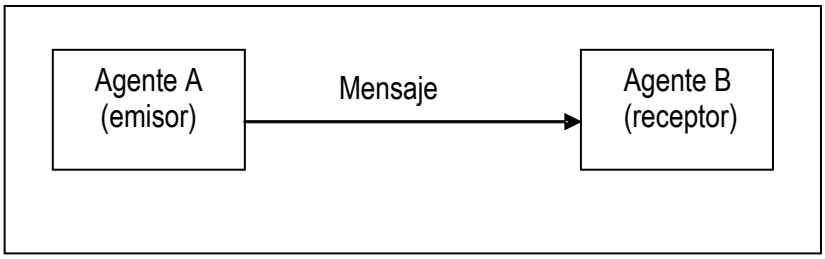

Ilustración 4: Principio de la transmisión de un mensaje

Para permitir que los mensajes sean utilizados para implementar estrategias de cooperación debemos aclarar dos cuestiones. Primero, debemos definir un protocolo de comunicaciones que especifique exactamente el proceso de comunicación, el formato de los mensajes y la elección del lenguaje de comunicaciones. Segundo, todos los agentes asociados deben conocer la semántica del lenguaje de comunicación.

Si analizamos la comunicación humana, determinamos que no sólo simples afirmaciones, verdadero o falso, pueden intercambiarse a través de la transferencia de mensajes, sino que los mensajes pueden iniciar acciones simples o cadenas completas de acciones. Por ejemplo, si se pregunta "¿Me puede dar el libro?", normalmente no se espera un simple si o no, sino que viene implícita la solicitud de entrega del libro. Se inicializa una acción concreta. Las intenciones del emisor no son siempre tan obvias como en este ejemplo, sino que el contenido del mensaje depende del estado mental del emisor. Lo mismo ocurre con el receptor. Una simple petición puede causar respuestas completamente diferentes dependiendo del estado mental del receptor y de su relación con el emisor. La teoría de los actos de habla intenta atender estas cuestiones.

2.4 Niveles y Protocolos de Comunicación

Los protocolos de comunicación están normalmente especificados en diferentes niveles. El nivel inferior del protocolo especifica el método de interconexión; el nivel medio especifica el formato o sintaxis de la información que es transferida; el nivel superior especifica el significado o semántica de la información. La semántica no solo se refiere a la esencia del mensaje, sino también al tipo de mensaje.

Existen protocolos de comunicación binarios y n-arios. Un protocolo binario involucra a un único emisor y a un único receptor, mientras que un protocolo n-ario involucra a un único emisor y a múltiples receptores (llamado "broadcasting" o "multicasting"). Un protocolo está especificado por una estructura de datos con los siguientes cinco campos:

1. Emisor
2. Receptor (o receptores)
3. Lenguaje en el protocolo
4. Funciones de codificación y decodificación
5. Acciones realizadas por el receptor o los receptores

Una decisión fundamental para la interacción de agentes es separar la semántica del protocolo de comunicación (independiente del dominio) de la semántica del mensaje encapsulado (dependiente del dominio). El protocolo de comunicaciones debe ser conocido por todos los agentes. Debe ser conciso y tener un número limitado de actos de comunicación primitivos.

Lo visto hasta este momento sugiere la necesidad de un lenguaje que permita la intercomunicación entre nuestros agentes autónomos distribuidos (ACL "Agent Communication Language"). Existen principalmente dos propuestas que se están convirtiendo en especificaciones estándares que siguen la mayoría de sistemas de desarrollo de agentes. Una de ellas está basada en la utilización del lenguaje de comunicación KQML. Por otro lado hay otro grupo de investigadores que siguen las descripciones dadas por la organización FIPA. Veamos con más detalle estas propuestas.

2.4.1 Especificaciones FIPA

FIPA ("Foundation for Intelligent Physical Agents") [Url_FIPA] fue creada en 1996 para producir software estándar para sistemas basados en agentes, heterogéneos e interactuando entre ellos. FIPA ha elaborado una serie de especificaciones que pueden ser usadas para el desarrollo de sistemas basados en agentes.

Las distintas especificaciones FIPA van progresando a través de un ciclo de vida (ver Ilustración 5), desde un estado *preliminar* (borrador en construcción cuyo identificador empieza por P), *experimental* (especificación estable que sólo admite pequeños cambios, cuyo identificador empieza por X) hasta el estado *estándar* (especificación implementada con éxito en varias plataformas; su identificador empieza por S). Cualquier especificación puede ser *reprobada* (el identificador empieza con D), previo al estado de *obsoleta* (su identificador empieza por O), cuando se considera innecesaria por cambios en la tecnología o en otras especificaciones y ha quedado por tanto obsoleta.

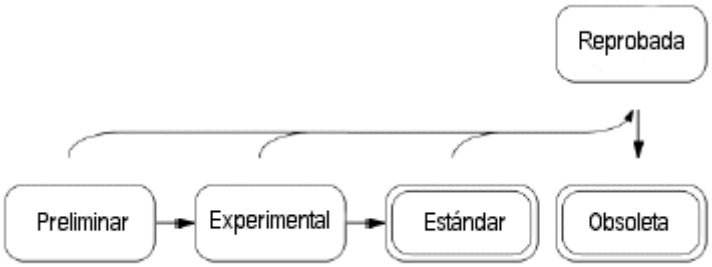

Ilustración 5: Ciclo de vida de las especificaciones

Las especificaciones FIPA se pueden organizar en cinco materias, según el dominio de agentes al que va dirigido:

- Aplicaciones
- Arquitecturas Abstractas
- Comunicación entre Agentes
- Gestión de Agentes
- Transporte de Mensajes de los Agentes

En este apartado vamos a centrarnos únicamente en las especificaciones referentes a los mensajes escritos en un Lenguaje de Comunicación entre Agentes. Estas a su vez están organizadas en *Protocolos de Interacción* para el intercambio de mensajes, *Actos Comunicativos* basados en la teoría de los actos de habla y *Lenguajes de Contenido* (ver Ilustración 6).

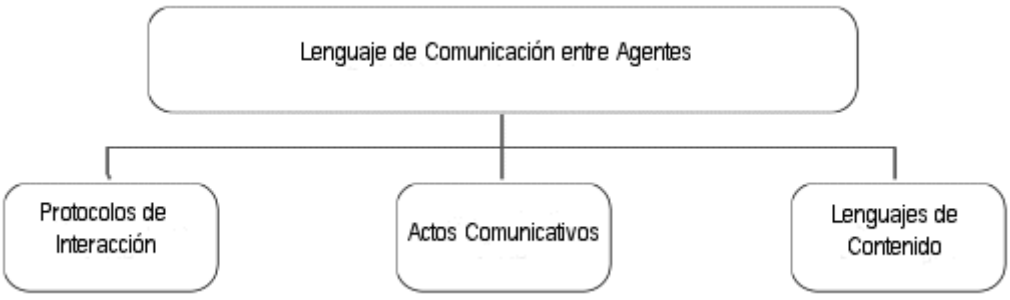

Ilustración 6: Especificaciones de ACL

Tabla 1: Elementos de un mensaje en ACL de FIPA

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

FIPA propone distintas especificaciones para representar el contenido de los mensajes en distintos lenguajes: SL ("Semantic Language"), CCL ("Constraint Choice Language"), KIF ("Knowledge Interchange Format") y RDF ("Resource Descripction Framework").

2.4.2 Especificaciones KQML

KQML ("Knowledge Query and Manipulation Language") [Url_KQML], [Finin,1994], [Labrou,1997] es un lenguaje y protocolo para el intercambio de información y conocimiento, surgido de los trabajos para el desarrollo de técnicas y metodologías para la construcción de bases de conocimiento a gran escala que sean compartidas y reusables. La sintaxis de KQML consiste en una notación en forma de listas de paréntesis balanceados (similar a LISP). KQML es un lenguaje basado en la teoría de actos de habla. Fue concebido como un formato de mensajes y como un protocolo que maneja los mensajes para permitir a un programa identificar, conectarse e intercambiar información con otros programas. En términos lingüísticos se puede decir que KQML se enfoca principalmente a la parte pragmática de la comunicación. Puede ser usado como un lenguaje para programas de aplicación que interactúen con un sistema inteligente o para dos o más sistemas inteligentes que compartan conocimiento en la

resolución cooperativa de problemas. Y, de momento, es uno de los modelos para comunicaciones entre agentes más usado.

Tres características importantes de KQML son:

1. Los mensajes de KQML son opacos al contenido de lo que transportan, esto es, los mensajes en KQML no comunican únicamente oraciones en un lenguaje, sino que comunican una *actitud* acerca del contenido (por ejemplo, afirmación, solicitud, pregunta).
2. Las primitivas del lenguaje, que se llaman *performativas*, indican las acciones u operaciones permitidas que los agentes pueden utilizar cuando se comunican (actos de habla).
3. El entorno en el que los agentes se comunican con KQML puede ser enriquecido con un tipo de agentes especiales llamados *facilitadores*, que son una clase especial de agentes que coordinan las interacciones entre los otros agentes.

KQML es un lenguaje que se divide en tres capas o niveles en los cuales codifica la información (Ilustración 7):

- La *capa de contenido* se relaciona con el contenido real del mensaje escrito en el lenguaje de representación propio de cada agente. Un mensaje en KQML puede tener cualquier lenguaje de representación incluyendo lenguajes expresados como cadenas en ASCII y aquellos expresados utilizando una notación binaria. Cualquier implementación de KQML ignora la parte del contenido del mensaje excepto para determinar dónde termina.
- La *capa de comunicación* codifica un conjunto de características del mensaje, las cuales describen los parámetros de la comunicación de bajo nivel, tales como la identidad del agente que envía el mensaje y la del que lo recibe, así como un identificador único asociado con la comunicación.
- La *capa de mensaje* se utiliza para codificar el mensaje que una aplicación desea transmitir a otra. Esta capa forma el corazón del lenguaje y determina las clases de interacciones que se pueden tener con un agente que hable KQML. La función principal de la capa de mensaje es identificar el protocolo que se va a usar para entregar el mensaje (síncrono o asíncrono) y proporcionar una performativa que el transmisor le agrega al contenido. Además de esto, ya que el contenido es opaco a KQML, en esta capa también se incluyen características opcionales que describen el lenguaje del contenido, la ontología que se está asumiendo y alguna clase de descripción del contenido. Estas características hacen posible que las implementaciones de KQML analicen, redirigan y entreguen el mensaje apropiadamente aún cuando su contenido sea inaccesible.

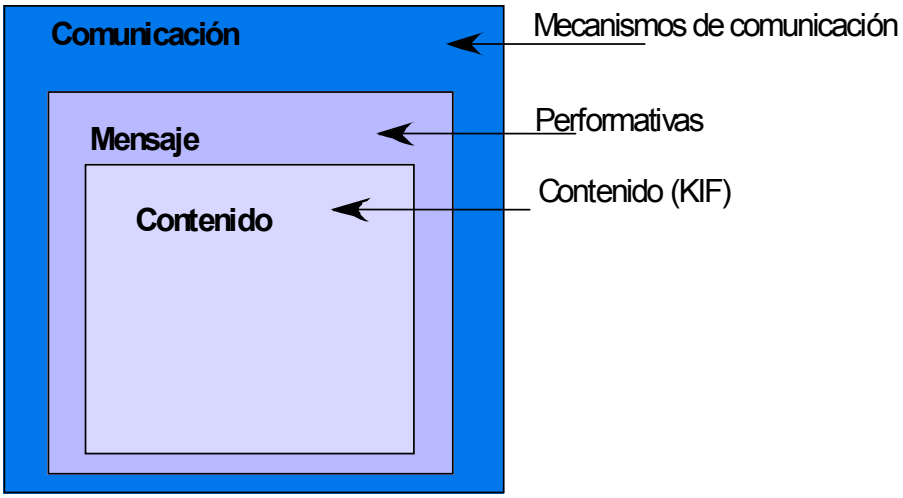

Ilustración 7: Capas del lenguaje KQML

Todo mensaje en KQML se inicia con una directiva de comunicación (performativa o *expresión realizativa*) que indica el tipo de comunicación y un conjunto de argumentos opcionales, llamados *parámetros*, entre los que se encuentra el contenido real del mensaje así como otros que describen el propio contenido.

Existe un conjunto de performativas estándar con un significado al que toda implementación de KQML debe adherirse. Estas performativas estándar se dividen en tres grupos:

- Las performativas *de discurso* (`ask-if`, `ask-all`, `ask-one`, `tell`, `deny`, `achieve`, `advertise`, `subscribe`, entre otras), empleadas en el contexto de un intercambio de información y conocimiento entre dos agentes.
- Las performativas *de intervención* y *mecánica de la conversación* (`error`, `sorry`, `ready`, `next`, `discard`, `rest`, `standby`), cuyo papel es intervenir en el curso normal de una conversación.
- Las performativas *de red* y *de facilitación* (`register`, `forward`, `broadcast`, `recommend-one`, `recruit-all`, entre otras), los cuales, estrictamente hablando no son actos del habla, pero permiten a los agentes encontrar otros agentes capaces de procesar sus mensajes.

Los nombres de los parámetros empiezan con ":" y deben estar seguidos por el correspondiente *valor del parámetro*. Los posibles parámetros con su correspondiente descripción se pueden ver en la siguiente tabla.

Tabla 2: Parámetros reservados para las performativas KQML

Palabra clave	Descripción
:sender	emisor (actual) de la performativa
:receiver	receptor (actual) de la performativa
:from	el origen (emisor virtual) de la performativa de :content cuando se hace un forward
:to	el destinatario final (receptor virtual) de la performativa de :content cuando se hace un forward
:in-reply-to	etiqueta esperada en la respuesta al mensaje previo (la misma de :reply-with del mensaje previo)
:reply-with	etiqueta esperada en la respuesta al actual mensaje
:language	lenguaje de representación en el que está escrito :content
:ontology	nombre de la ontología asumida en :content
:content	información con la que la performativa expresa una actitud

El protocolo básico está definido por la siguiente estructura:

```
(<KQML-performativa>
      :sender     <nombre>
      :receiver   <nombre>
      :languaje   <nombre>
      :ontology   <nombre>
      :content    <expresión>
      ...
)
```

El siguiente es un ejemplo de un mensaje KQML donde un agente que se identifica como `jorge` solicita a un `facilitador` que envíe un mensaje proveniente del mismo `jorge` y escrito en KQML a otro agente `ana`, asumiendo la ontología `ontologia-fdl`. A su vez el mensaje que `jorge` quiere hacer llegar a `ana` a través de `facilitador` indica que `jorge` desea que `ana` haga cierto en su entorno lo que está en el contenido del mensaje (`estado=suspendido`) y que le responda con un mensaje identificado por `id1`. El parámetro `content` define el nivel de contenido; los parámetros `from`, `to`, `sender` y `receiver` especifican información del nivel de comunicación; y `language` y `ontology` forman parte del nivel de mensaje.

```
(forward
      :from jorge
      :to ana
      :sender jorge
      :receiver facilitador
      :language kqml
      :ontology ontologia-fdl
      :content (achieve
            :sender jorge
```

```
                :receiver ana
                :reply-with id1
                :content (estado=suspendido)
        )
    )
```

Intuitivamente, podemos decir que cada mensaje en KQML es una pieza de diálogo entre emisor y receptor, y KQML proporciona el soporte para una amplia variedad de tipos de diálogos. El conjunto de performativas es extensible y no todas son necesarias. Un agente puede elegir manejar sólo unas pocas performativas de las descritas, e incluso implementar algunas adicionales. Sin embargo, cualquier implementación de las performativas reservadas debe ser utilizada de la manera descrita.

2.5 Semántica y Ontologías

El objetivo de la ontología es el estudio de todas las clases de entidades que conforman nuestro mundo, de la existencia en sí. Así, llamaremos una *ontología* a un catálogo de los tipos de cosas que asumimos que existen en un domino de interés, desde la perspectiva de una persona que usa un lenguaje con el propósito de hablar acerca de ese dominio. Aunque el término procede de la Filosofía (ontología: parte de la filosofía que estudia el ente en cuanto tal), en los últimos tiempos ha sido reutilizado en el campo de la Inteligencia Artificial y en concreto en los Sistemas Multiagente. Así podemos definir la ontología como [Neches,1991]:

Una ontología define los términos y relaciones básicas que forman parte del vocabulario de una determinada área, así como las reglas para combinar términos y relaciones para definir extensiones del vocabulario

Los elementos de una ontología son [Gómez,1999]:

- Conceptos: cualquier cosa sobre la que podamos emitir un juicio o comentario.
- Relaciones: representan diferentes tipos de interacción entre conceptos del dominio.
- Funciones: son un caso especial de relaciones.
- Instancias: representan cada uno de los elementos.
- Axiomas: sentencias que son siempre verdaderas en ese dominio.

Las dos principales fuentes de categorías ontológicas son:

- La observación, que proporciona conocimiento del mundo físico.
- El razonamiento, que da sentido a las observaciones generando un marco de abstracción llamado metafísica.

La ontología define las clases de cosas que existen en el dominio de aplicación, evitando que los términos y símbolos estén mal definidos o sean confusos. La elección

de las categorías ontológicas es el primer paso en el diseño de una base de conocimiento. Esta selección de categorías determinará todas las cosas que pueden ser representadas en nuestro sistema de agentes. Un agente debe representar sus conocimientos en el vocabulario de una ontología específica. El creador del agente debe utilizar una ontología específica para representar el conocimiento del agente.

El vocabulario del lenguaje empleado para comunicarse los agentes consiste en un diccionario de palabras apropiado para áreas de aplicación comunes. Cada palabra en el diccionario tiene una descripción (escrita en lenguaje natural) que es usada por las personas para entender su significado y una anotación formal (por ejemplo, escrita en KIF) que es usada por los programas. El diccionario es abierto, es decir, es posible añadir nuevas palabras dentro de áreas existentes y en nuevas áreas de aplicación. La existencia de este diccionario no significa que solamente hay una manera de describir un área de aplicación. Un diccionario puede contener múltiples ontologías para un área dada y un agente puede utilizar la ontología que le sea más conveniente. Las definiciones formales asociadas con cualquiera de estas ontologías pueden ser utilizadas por los agentes para traducir mensajes que usan una ontología en específico a mensajes que usan otras ontologías. Cuando se comparte información en una comunidad se debe tener un acuerdo sobre el significado de los símbolos. Si esta comunidad es pequeña es posible que todos los objetos que se comunican utilicen un vocabulario único. Sin embargo, si la comunidad es extensa, es muy posible que sea necesario soportar una colección de vocabularios.

2.6 Comunicación eficiente

La sabiduría popular dice que "a buen entendedor, pocas palabras bastan". Para conseguir una comunicación más eficiente (menos mensajes y más cortos) podemos asumir que el agente utilizará su propio conocimiento para determinar el significado de una sentencia. Esta es una de las razones por la que a los ordenadores les resulta muy difícil comprender el lenguaje natural. Para comprender una conversación de dos personas, necesitamos conocimiento sobre el contexto en el que se desarrolla la comunicación y conocimiento general (*conocimiento ingenuo* o de *sentido común*) compartido por los dos interlocutores. Algunos aspectos a tener en cuenta para una comunicación eficiente son:

- Utilización del contexto: si el oyente y el hablante comparten un mismo contexto, pueden utilizarlo como fuente de conocimiento para determinar el significado de las expresiones. Gracias al contexto podemos incluir en el lenguaje pronombres y referencias.

- Resolución de ambigüedades: en el campo de la lingüística computacional se han identificado varios tipos de ambigüedades que pueden aparecer en las expresiones en lenguaje natural:
    - Ambigüedad léxica: una misma palabra puede tener varios significados.
    - Ambigüedad sintáctica: existen oraciones que pueden ser analizadas de distintas formas.
    - Ambigüedad referencial: la utilización de pronombres y anáforas puede provocar ambigüedades.
    - Ambigüedad pragmática: si el conocimiento de sentido común y el conocimiento sobre el contexto que posee el oyente no es preciso, puede que no se puedan resolver algunas ambigüedades.

2.7 Negociación

La *negociación* es importante en la modelización de sistemas multiagente. Podemos considerarla como un proceso de comunicación entre un grupo de agentes para cumplir un contrato mutuamente aceptado. La negociación en los sistemas multiagente se observa desde varios puntos de vista. Por una parte, tanto la asignación de subproblemas como la asignación de recursos puede contemplarse como una negociación; por otro lado, también deben existir negociaciones entre agentes individuales para desarrollar una cooperación en beneficio del sistema. La negociación puede ser *competitiva* o *cooperativa*, según el comportamiento de los agentes individuales.

Existen varias situaciones diferentes a la hora de entablar una negociación:

- Ninguno de los agentes obtiene beneficio a través de la negociación, puesto que cada agente sigue su propio objetivo y no existen dependencias directas entre ambos objetivos.
- Al menos uno de los agentes logra el objetivo de forma más rápida o con menos esfuerzo.
- Existen situaciones reales de conflicto entre los agentes, debido a recursos, logros de objetivos similares, etc.

Una clasificación interesante desde el punto de vista individual del agente divide en cuatro las posibles formas de actuar en la negociación:

- *Cooperación simétrica*. La negociación produce resultados mejores que los que cada agente obtendría de manera individual.
- *Compromiso simétrico*. Los agentes logran el objetivo independientemente y negocian un compromiso entre ambas partes, de manera que se degrada el resultado en comparación con las soluciones independientes. Es esencial un compromiso debido a que no se puede obviar a ninguno de los agentes.

- *Cooperación/compromiso no simétrico*. Uno de los agentes obtiene una mejora de su resultado, mientras que el otro empeora su solución.
- *Conflicto*. No se puede llegar a un acuerdo razonable debido a que los objetivos entran en conflicto. La negociación debe terminar sin obtener un resultado claro.

En las estrategias de negociación, actualmente están tomando relevancia los algoritmos genéticos, puesto que cada agente podría generar un número aleatorio de estrategias de negociación y aplicarlas, utilizando posteriormente aquellas que den mejor resultado. Sin embargo, el sistema podría requerir de un número grande de estrategias de negociación, con lo que puede resultar inviable. También está creciendo el uso de la teoría de juegos en el ámbito de la negociación, asumiendo negociaciones más bien agresivas. Los conceptos claves de teoría de juegos aplicada a la negociación son:

- *Función de Utilidad*: la *utilidad* puede ser definida como la diferencia entre el *beneficio* de conseguir un objetivo y el *coste* invertido para lograrlo.
- *Espacio de Transacciones*: una *transacción* es una acción que un agente hace y que lleva una utilidad asociada.
- *Estrategias y Protocolos de Negociación*: un protocolo de negociación define las reglas que gobiernan la negociación, incluyendo cómo y cuando finaliza la misma.

Las características más importantes de una negociación son el lenguaje utilizado por los agentes participantes, el protocolo seguido por los agentes para la negociación y el proceso de decisión (estrategias) que cada agente utiliza para determinar su posición, concesiones y criterios para el acuerdo. Un mecanismo de negociación debe tener las siguientes características:

- *Eficiencia*: los agentes no deben gastar recursos para llegar a un acuerdo.
- *Estabilidad*: ningún agente debe tener incentivos para desviarse de las estrategias de acuerdo.
- *Simplicidad*: los mecanismos de negociación deben imponer demandas de computación y de ancho de banda bajas a los agentes.
- *Distribución*: el mecanismo no debe requerir de un sistema de decisión central.
- *Simetría*: el mecanismo no debe tener ningún prejuicio sobre un agente por razones arbitrarias o inapropiadas.

La negociación automática puede ahorrar tiempo a los humanos a demás de realizar contratos más beneficiosos [Rosenschein,1994]. Existen distintas técnicas de negociación: votación (“voting”), subasta (“auction”), pacto (“bargaining”), mecanismos de mercado (“market mechanisms”), contratación (“contracting”) y coaliciones (“coalition formation”).

3 Arquitecturas de Agente

Existen muchos tipos de agentes inteligentes, cada uno de ellos desarrollando las tareas para las que ha sido creado. Y mientras para unos el tiempo se convierte en un recurso crítico, otros podrán tomar decisiones más racionales al disponer del tiempo y el conocimiento suficiente para ello; unos agentes serán más complejos e inteligentes que otros. No existe, por tanto, una única arquitectura ideal para agentes inteligentes. La estructura concreta de las arquitecturas dependerá de las tareas y el entorno donde éstas se desarrollen. Vamos a analizar algunas arquitecturas que han servido de referencia y que pueden ser representativas del amplio abanico de posibilidades que existe.

3.1 Arquitecturas reactivas

Las arquitecturas reactivas se basan en una estrecha relación entre percepción y acción. Funcionan bien en entornos de tiempo real ya que son computacionalmente económicas.

Los agentes reactivos no poseen un modelo simbólico de su entorno. La capacidad de realizar complejos procesos de razonamiento también está omitida. La razón de estas restricciones está en la creación de agentes compactos, tolerantes a fallos, y sobre todo, flexibles. Al contrario que otros tipos de agentes, los agentes reactivos no obtienen su inteligencia de modelos internos sino de una interacción con su entorno. Por lo tanto, la importancia del proceso de interacción es alta. Los que defienden la estructura reactiva defienden que la inteligencia no existe en sistemas individuales, sino que es parte implícita de un entorno completo (propiedad *emergente*).

Un agente reactivo no debe tener necesariamente una estructura compleja para ser capaz de actuar en un entorno complejo. Basta con que observe el entorno y reconozca una serie de principios simples o dependencias. Este conocimiento es utilizado para desarrollar módulos específicos que sean capaces de comprobar continuamente en su entorno la ocurrencia de situaciones específicas y de iniciar una reacción directa cuando dichas situaciones tienen lugar.

La siguiente figura (Ilustración 8) muestra la arquitectura fundamental de los agentes reactivos que se corresponden con un sistema simple de estímulo/respuesta. Los *sensores* recogen la información, la envían a los *módulos de competencia* correspondientes, produciéndose una reacción como salida en los mismos, que se transmite al exterior por medio de *actuadores*.

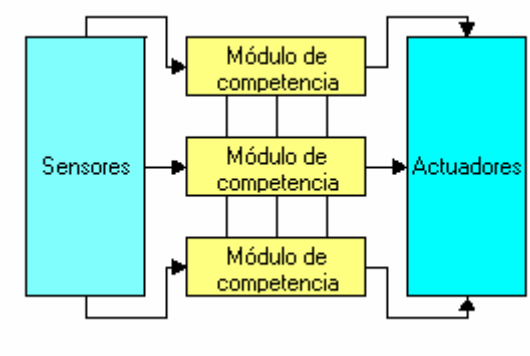

Ilustración 8: Arquitectura de agentes reactivos

Los robots clásicos son un ejemplo simple de agentes reactivos. Un robot posee un determinado número de sensores que le permiten observar su entorno. Por ejemplo, puede determinar cuando ha chocado con un obstáculo, que objetos han cambiado de posición o si otro robot está activo en su vecindad. Esta información es pasada a los módulos de competencia del robot. Por ejemplo, un módulo con la tarea de cambiar la dirección de movimiento está activo siempre que se detecte un obstáculo en el camino del robot. Los actuadores en este caso son las ruedas delanteras del robot, cuyo alineamiento cambia según las órdenes del módulo correspondiente.

En general, los sensores de un agente reactivo le permiten obtener información de su entorno y reconocer la ocurrencia de situaciones cambiantes. La forma específica de los sensores (y también de los actuadores) depende en gran medida de los objetos monitorizados por el sensor.

La información recogida por el sensor es transmitida al módulo de competencia correspondiente. Esto normalmente se hace en un formato de texto plano, es decir, no se utilizan representaciones simbólicas o lenguajes de comunicación de alto nivel. Cada módulo de competencia está encargado de una tarea claramente definida pero no particularmente compleja. Un agente de información, por ejemplo, puede tener módulos para buscar fuentes de información, recoger los resultados de la búsqueda y presentar dichos resultados. Las propiedades necesarias para llevar a cabo estas tareas están contenidas dentro de los módulos correspondientes. No hay componentes centrales, como un planificador o un razonador. Todos y cada uno de los módulos debe poseer todas las capacidades requeridas para realizar su tarea. Como consecuencia, un agente reactivo no puede resolver una tarea para la que no tenga un módulo de competencia.

Los módulos trabajan en paralelo, pudiéndose comunicar directamente entre ellos o por medio del entorno. La comunicación directa toma la forma de una relación uno a uno entre dos componentes. Se basa en mecanismos de comunicación simples y no en un lenguaje complejo. La ventaja principal de este método es la rápida capacidad de reacción. El segundo método de comunicación se produce cuando un módulo produce

un cambio en su entorno que es observado por otro módulo, que inicia su proceso de respuesta. A pesar de que este método de comunicación es más lento, permite reaccionar a situaciones más complejas. Al trabajar los módulos de competencia en paralelo, varios de estos módulos pueden dispararse simultáneamente. Debe existir un mecanismo para elegir entre las distintas acciones seleccionadas. Brooks propone organizar los módulos en una jerarquía, situándose estos módulos en capas. Las capas inferiores de la jerarquía pueden inhibir a las capas superiores; cuanto más bajo sea el nivel de una capa, más prioritaria es.

La estructura descentralizada de los módulos de competencia incrementa la tolerancia a fallos y la robustez. Si un módulo falla o funciona incorrectamente seguramente el resto podrán seguir desempeñando sus tareas.

Los agentes reactivos normalmente no poseen la capacidad de desarrollar planes. Esto significa que sus tareas no pueden ser completamente determinadas. A pesar de que los planes pueden ser utilizados para optimizar el comportamiento de los agentes, hay factores, especialmente en aplicaciones prácticas, que hablan en contra del procesamiento de planes optimizados. Por ejemplo, un agente normalmente posee información incompleta de su entorno y sus propios objetivos. Su entorno es altamente dinámico y sujeto a continuos cambios; sus objetivos cambian dinámicamente, y los recursos necesarios para la creación de planes óptimos no suelen estar disponibles.

Otro punto de discusión es si los agentes reactivos son capaces de mostrar un comportamiento orientado a los objetivos. Los módulos de competencia tienen objetivos implícitos definidos por su funcionalidad que no se pueden cambiar. No pueden usar conocimiento interno para generar y seguir dinámicamente nuevos objetivos. Sin embargo, a pesar de estas restricciones, algunas personas tienen la opinión de que los agentes reactivos son capaces de acciones orientadas al objetivo. Similarmente a la inteligencia, la orientación hacia los objetivos resulta implícitamente de la interacción con el entorno, y no de la evaluación centralizada o la planificación. Por ejemplo, cada modulo de competencia del agente sigue un objetivo implícito que resulta de su tarea. Estos objetivos siempre se vuelven aparentes cuando la interacción es realizada con los objetos del entorno.

3.2 Arquitecturas Deliberativas

Las arquitecturas deliberativas siguen la corriente de la IA simbólica, que se basa en la hipótesis de los sistemas de símbolos-físicos enunciada por Newell y Simons, según la cual un sistema de símbolos físicos capaz de manipular estructuras simbólicas puede exhibir una conducta inteligente. Para poder trabajar en el nivel de Conocimiento de Newell, nuestro problema será cómo describir los objetivos y medios de satisfacerlos, y cómo realizar la traducción del nivel de conocimiento al nivel simbólico.

Las decisiones se toman empleando mecanismos deductivos:

Emparejamiento de patrones ("Pattern matching").

Diversos formalismos lógicos.

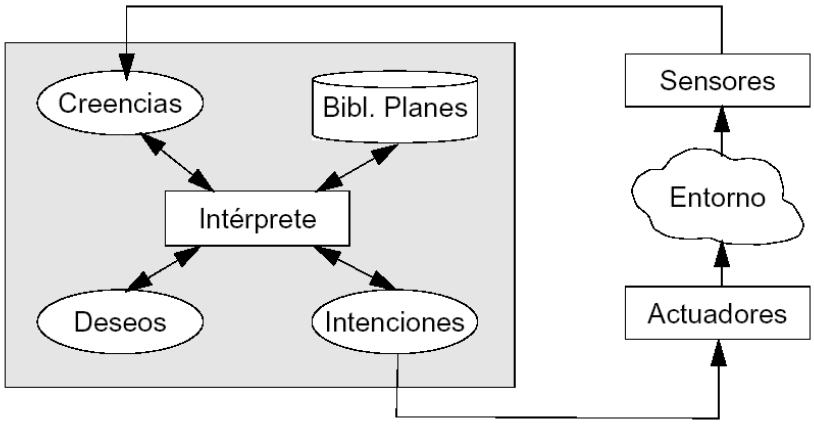

Ilustración 9: Arquitectura de agentes deliberativos

Las arquitecturas de agentes deliberativos (Ilustración 9) suelen basarse en la teoría clásica de planificación de inteligencia artificial: dado un estado inicial, un conjunto de operadores/planes y un estado objetivo, la deliberación del agente consiste en determinar qué pasos debe encadenar para lograr su objetivo, siguiendo un enfoque descendente ("top-down"). Como ejemplo de arquitectura cuyo componente principal es un planificador podemos citar los *Softbots*, cuya misión consiste en ayudar a los usuarios a realizar las tareas típicas de Unix.

Podemos distinguir los siguientes tipos principales de arquitecturas deliberativas o simbólicas: arquitecturas intencionales y arquitecturas sociales.

Los *agentes intencionales* se distinguen por ser capaces de razonar sobre sus creencias e intenciones. Se pueden considerar como sistemas de planificación que incluyen creencias e intenciones en sus planes.
Los *agentes sociales* se pueden definir como agentes intencionales que mantienen además un modelo explícito de otros agentes y son capaces de razonar sobre estos modelos.

3.3 Arquitecturas basadas en la Lógica

En las arquitecturas basadas en la lógica, la toma de decisiones es vista como una deducción lógica. Tienen las ventajas de la claridad semántica y el poder aprovechar todos los avances y desarrollos de la lógica y la demostración automática. Tienen la desventaja de que las arquitecturas puramente lógicas no son adecuadas cuando tenemos restricciones de tiempo (respuesta en tiempo real).

Los agentes suelen ser utilizados en entornos de producción complejos, donde el fallo o un comportamiento incorrecto del agente puede ser negativo. Por lo tanto, se busca desarrollar técnicas que aseguren que los agentes se comportarán de una forma determinada. Trabajos previos en Ciencia de la Computación han estudiado los métodos formales como una base para crear sistemas con un número mínimo de errores. La aproximación tradicional para construir sistemas de inteligencia artificial (IA simbólica) sugiere que un comportamiento inteligente puede generarse dándole al sistema una representación simbólica del entorno y del comportamiento deseado, y manipulando sintácticamente esta representación. La representación simbólica del entorno se consigue mediante fórmulas lógicas, y la manipulación sintáctica, a través de una deducción lógica o demostración de teoremas.

En tales agentes, se asume que el estado es una base de datos de fórmulas de predicados de lógica de primer orden. La base de datos es la información que tiene el agente sobre el entorno. El proceso de toma de decisiones del agente se modela a través de un conjunto de reglas de deducción (reglas de inferencia). Así, el comportamiento del agente queda determinado por las reglas de deducción y la base de datos actual.

Las aproximaciones basadas en la lógica son elegantes y tienen una semántica clara (la semántica del formalismo lógico). Pero tienen varias desventajas:

> No son instantáneos: la complejidad computacional inherente a la demostración de teoremas hace cuestionable si los agentes basados en lógica o deliberativos pueden operar eficazmente en entornos con restricciones temporales. La toma de decisiones de este tipo de agentes se basa en la suposición de un entorno que no cambia significativamente mientras el agente está decidiendo qué hacer, y en que la acción, que era racional en el momento que comenzó el proceso de decisión, sea racional cuando éste concluya.
> Representación del entorno: este tipo de agentes utiliza una representación simbólica del entorno que típicamente es un conjunto de fórmulas en el lenguaje de representación agente. Para entornos complejos, dinámicos y posiblemente físicos, este tipo de correlación no es obvia.

Razonamiento sobre el entorno: hasta la representación del conocimiento procedimental simple puede no ser intuitiva y laboriosa en la lógica tradicional. El razonamiento sobre la información temporal (cómo cambia una situación en el tiempo) resulta ser extremadamente difícil. Este es el caso de entornos complejos, dinámicos y físicos.

La formalización de los sistemas de agentes ha sido utilizada con distintos objetivos:

Como un lenguaje de especificación interno usado por el agente para su razonamiento.
Como metalenguajes externos usados por el diseñador para especificar, diseñar y verificar ciertas propiedades del comportamiento de los agentes situados en un entorno dinámico.

La primera aproximación es la más tradicional. Presupone que los agentes poseen explícitamente la capacidad de razonar. Estos agentes son normalmente conocidos como cognitivos, racionales, deliberativos o pesados. La segunda aproximación es más reciente en el estudio de agentes, aunque es más tradicional en el resto de ciencias de la computación. Permite usar el formalismo para permitir al diseñador razonar sobre el agente. El agente puede o no ser capaz de razonar por sí mismo una vez en uso. Lo ideal sería disponer del mismo lenguaje lógico para los dos propósitos anteriores. Sin embargo, los requisitos de los agentes situados en entornos dinámicos requieren que el lenguaje interno sea computacionalmente eficiente, mientras que la variedad de complejos comportamientos que son posibles en un sistema de agentes autónomos requiere que el lenguaje externo sea más expresivo.

Se suelen utilizar los siguientes tipos de lógicas:

Lógica proposicional y *de predicados*: es la más simple y la más usada. Las fórmulas se construyen a partir de proposiciones atómicas, que expresan hechos atómicos, y conectivas lógicas, que denotan “y”, “o”, “no” e “implica”. Las proposiciones atómicas y sus combinaciones mediante conectivas son usadas para describir los estados del sistema. No consideran como el sistema ha evolucionado o lo está haciendo

Lógica modal: ha sido utilizada para estudiar los diferentes modos de verdad, como *posiblemente* cierto y *necesariamente* cierto. En el estudio de agentes, es usado para dar significado a conceptos como creencia y conocimiento. La lógica proposicional es extendida con dos operadores, el de posibilidad y el de necesidad.

Lógica deóntica: permite indicar lo que un agente esta obligado a hacer. La lógica deóntica tradicional introduce un operador de obligatoriedad. Se especifica como una lógica modal en la que el axioma principal es Obl p -> no Obl no p (el agente es obligado a p solo si no está obligado a no p).

Lógica dinámica: puede ser vista como la lógica modal de la acción. Los operadores de necesidad y posibilidad están basados en el tipo de acciones disponibles. Las acciones son expresadas mediante expresiones regulares, siendo las acciones atómicas aquellas que el agente puede llevar a cabo directamente. La semántica es la de la lógica modal pero incluyendo un conjunto de estados o mundos definidos por posibles transiciones basados en las acciones.

Lógica temporal: hay diversas variantes. Las distinciones más importantes son las siguientes:

- Lineal contra ramificada: según el tiempo sea visto como un simple curso de la historia o varios posibles cursos de la historia. La ramificación puede ser en el pasado, en el futuro o en ambos.
- Discreta contra densa: según el tiempo se vea como consistente en pasos discretos (como los números naturales) o teniendo siempre estados intermedios (como los números reales).
- Basada en momentos contra basada en periodos: según los átomos de tiempo sean puntos o intervalos.

3.4 Arquitecturas BDI

En las arquitecturas BDI ("Belief-Desire-Intention"), las tomas de decisiones se realizan sobre un proceso de razonamiento que parte de las creencias que el agente tienen del mundo y teniendo en cuenta las intenciones y las acciones.

Los componentes básicos de esta arquitectura son las creencias ("belief"), los deseos ("desire") y las intenciones ("intention") del agente; las funciones que representan su deliberación; y el razonamiento de fines y medios. Este tipo de arquitectura tiene sus raíces en la tradición filosófica del entendimiento del razonamiento práctico, que es el proceso de decidir, momento a momento, qué acción ejecutar para cumplir con los objetivos fijados.

El razonamiento práctico involucra dos procesos importantes: decidir qué metas se desean conseguir, proceso que se conoce como deliberación; y cómo se lograrán estas metas, procedimiento que se denomina razonamiento de fines y medios ("means-ends reasoning"). El proceso de decisión empieza típicamente tratando de comprender qué opciones están disponibles; una vez generado este conjunto de alternativas, se debe elegir entre ellas y comprometerse ("commit") con una; esta opción escogida se convierte en una intención, la cual determina las acciones del agente. Las intenciones localizan el razonamiento práctico futuro del agente; cuando se tiene una intención en particular, se descartan todas aquellas opciones que sean inconsistentes con la intención.

Además, una vez adoptada una intención, el agente debe perseverar ("persist") en ésta, sólo debe rectificarla cuando la razón por la cual tenía la intención ha cambiado; o cuando el agente sabe con certeza que no podrá cumplir con ella.
Finalmente, las intenciones están estrechamente relacionadas con las creencias acerca del futuro. Cuando tiene una intención, el agente al menos debe creer que tiene una gran posibilidad de cumplir con ella.

Interprete BDI

```
InicializarEstado();

Repetir
            Opciones:=GeneradorDeObjetivos(colaDeEventos);
            OpcionesSeleccionadas:=Deliberar(Opciones);
            ActualizarIntenciones(OpcionesSeleccionadas);
            Ejecutar();
            ObtenerNuevosEventosExternos();
            BorrarObjetivosConseguidos();
            BorrarObjetivosImposibles();

FRepetir
```

Un problema clave en el diseño del razonamiento práctico de los agentes es el de adquirir un buen equilibrio entre los diferentes intereses. A veces, los agentes deberían abandonar algunas de sus intenciones (ya sea porque cree que son inalcanzables, porque ya las ha conseguido o porque la razón por la que tenía la intención ya no está presente). Esto conlleva a que el agente deba reconsiderar sus intenciones cada tanto; lo que presenta un dilema que esencialmente es el de crear un equilibrio entre el comportamiento pro-activo (intencionado) y el reactivo (condicionado a eventos):

> Un agente que no se detiene lo suficientemente seguido a reconsiderar sus intenciones, continuará pretendiendo cumplir con ellas, aún cuando esté claro que no las puede consumar o la razón por la cual las tiene ya no está presente.
> Un agente que está constantemente reconsiderando sus intenciones puede dedicar poco tiempo al trabajo necesario para conseguirlas y por lo tanto corre el riesgo de nunca cumplirlas.

La dinámica del entorno condiciona este equilibrio. Cuanto más dinámico sea el entorno, la habilidad de reaccionar a cambios modificando las intenciones se hace más importante. Las razones por las que este modelo es atractivo son: primero porque es

intuitivo y segundo porque se tiene un conocimiento informal de los términos creencias, deseos e intenciones; el método de decisión se parece al razonamiento práctico usado diariamente; segundo da una descomposición funcional clara, la cual indica qué clase de subsistemas se pueden requerir para construir el agente. La dificultad principal es, como siempre, la de saber como implementar eficientemente estas funciones.

3.5 Arquitecturas híbridas

Estas arquitecturas pretenden combinar aspectos deliberativos y reactivos, mediante la combinación de módulos reactivos con módulos deliberativos. Los módulos reactivos se encargan de procesar los estímulos que no requieren deliberación, mientras que los módulos deliberativos determinan que acciones deben realizarse para satisfacer los objetivos locales y cooperativos de los agentes.

3.6 Arquitecturas por Capas

En las arquitecturas por capas, la toma de decisiones esta particionada en un número de capas, cada una de las cuales trata con el entorno del agente a diferente nivel de abstracción.

Dado el requerimiento que un agente sea capaz de tener comportamientos reactivos y proactivos, una descomposición clara es crear subsistemas separados para tratar estos tipos de comportamientos diferentes. La idea lleva naturalmente a una clase de arquitectura en la cual varios subsistemas son jerarquizados en capas que interactúan entre sí.
Típicamente habrá dos capas para tratar con el comportamiento reactivo y pro-activo, respectivamente. Sin embargo, cuantas más capas haya, más útil es la topología de tales arquitecturas, por el flujo de información y control que hay entre ellas. En general, se pueden identificar dos tipos de flujos de control:

> *Capas horizontales*: Cada capa de software está directamente conectada a las entradas sensoriales y a las salidas actuadoras. En efecto, cada capa actúa por sí misma, como un agente, produciendo sugerencias de qué acción realizar.
>
> *Capas verticales*: Una capa se encarga de manipular las entradas sensoriales y las salidas actuadoras.

La ventaja de las arquitecturas en capas horizontales es su simplicidad conceptual: si se necesita que un agente presente *n* tipos diferentes de comportamientos, entonces hace falta implementar *n* capas diferentes. No obstante, debido a que las capas están cada una compitiendo con las otras para generar la acción sugerida, existe el peligro de que el comportamiento global del agente no sea coherente. Para asegurar que las arquitecturas en capas horizontales sean consistentes, generalmente incluyen una función de mediación, la cual decide qué capa tiene el "control" del agente en un instante determinado. La necesidad de este control centralizado es problemática ya que obliga al diseñador a considerar todas las interacciones posibles entre las capas. Éste también introduce un cuello de botella en el algoritmo de decisión del agente.

Las arquitecturas deliberativas pueden clasificarse como horizontales porque los estímulos recibidos del exterior son procesados en varias capas de diferente nivel de abstracción y al final el nivel superior decide qué acciones hay que llevar a cabo (y las realiza directamente o se lo indica a las capas inferiores).

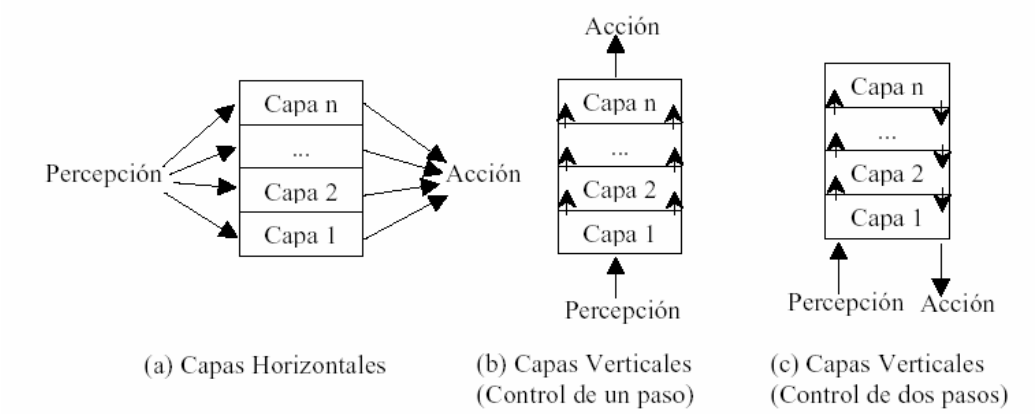

Ilustración 10: Arquitecturas por capas

Algunos de estos problemas se solucionan con la arquitectura en capas verticales. Éstas pueden dividirse en arquitecturas de capas verticales de un o dos pasos de control. En las primeras, el control fluye secuencialmente a través de cada capa, hasta que la capa final genera la acción de salida. En las segundas, la información fluye hacia los niveles superiores de la arquitectura mientras que el control fluye hacia las capas inferiores. En ambas, la complejidad de las interacciones entre las capas se reduce considerablemente. Sin embargo, esta simplicidad es a cambio de flexibilidad: para poder lograr que la arquitectura en capas verticales tome una decisión, el control debe pasar por cada una de las diferentes capas. Esto no es tolerante a fallos, y un fallo en cualquiera de las capas, trae graves consecuencias en el desempeño del agente.

3.6.1 "Touring Machines"

Este tipo de arquitectura organizada por capas horizontales está basada en tres capas productoras de actividades. Esto quiere decir que cada capa produce constantemente sugerencias sobre las acciones que el agente debería llevar a cabo. La *capa reactiva* provee de una respuesta más o menos inmediata a los cambios acontecidos en el entorno. Se implementa como un conjunto de reglas situación-acción.

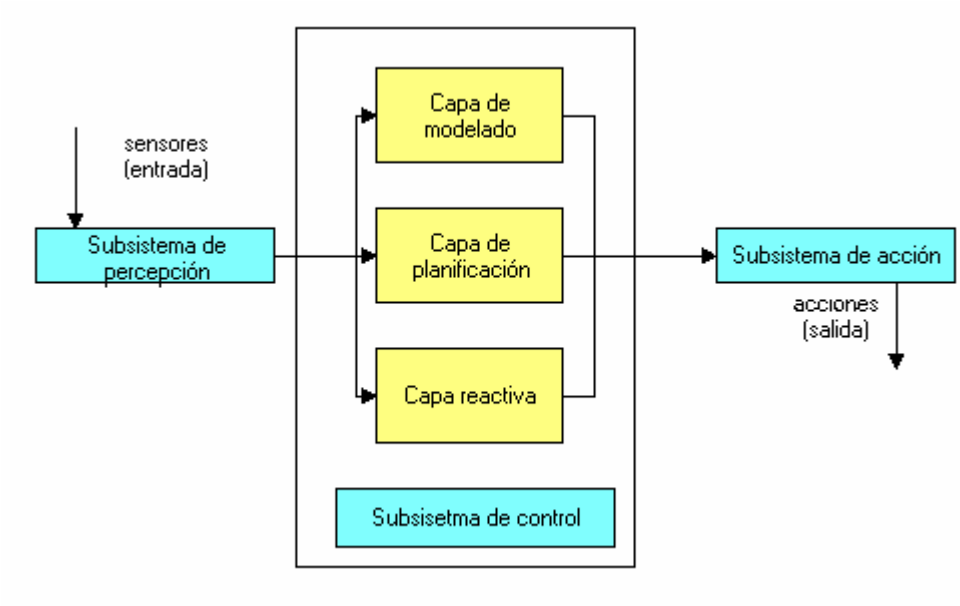

Ilustración 11: Arquitectura "Touring Machine"

La *capa de planificación* sirve para dotar de un comportamiento proactivo al agente. Bajo circunstancias normales, la capa de planificación es responsable de decidir qué hace el agente. Utiliza un conjunto de esqueletos de planes llamados *esquemas*. Estos esquemas son en esencia planes jerárquicamente estructurados que el agente elabora en tiempo de ejecución para decidir qué hacer. Para llevar a cabo un objetivo, la capa de planificación trata de encontrar un esquema que se corresponda a dicho objetivo. Este esquema contendrá sub-objetivos, que la capa de planificación utiliza para encontrar otros esquemas que se correspondan con los mismos.

La *capa de modelado* representa las diversas entidades del mundo (incluyendo el propio agente, así como otros agentes). Predice conflictos entre agentes y genera nuevos objetivos para resolver estos conflictos. Estos nuevos objetivos son pasados a la capa de planificación que busca los esquemas que los satisfagan.

Las tres capas de control están empotradas en un *subsistema de control*, que decide cual de las capas tendrá control sobre el agente. Este subsistema de control se implementa como un conjunto de reglas de control. Estas reglas de control pueden suprimir la información de los sensores para alguna capa o censurar las acciones de alguna capa.

3.6.2 "InteRRaP"

"InteRRaP" es un ejemplo de arquitectura de agente dividida verticalmente en capas y en dos pasos, tal como se ve en el siguiente diagrama:

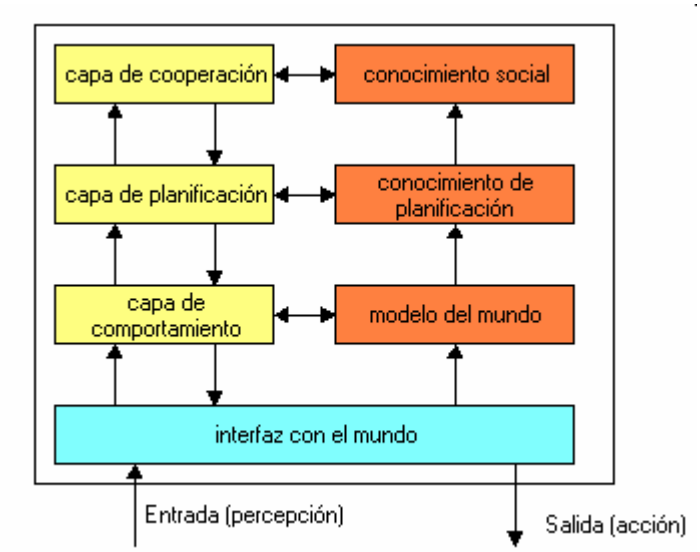

Ilustración 12: Arquitectura InteRRaP

Como en las "Touring Machines" se tienen tres capas de control. Además, el propósito de cada capa "InteRRaP" parece coincidir con la capa correspondiente de la "Touring Machine". Cada capa está asociada con una base de conocimientos, por ejemplo, una representación del mundo adecuada para cada capa. Estas bases de conocimiento representan al agente y su entorno a diferentes niveles de abstracción. La base de conocimientos del nivel más alto representa los planes y las acciones de los otros agentes en el entorno; la base de conocimiento del nivel medio representa los planes y acciones del propio agente; y la base de conocimiento del nivel más bajo representa información sobre el entorno. La explícita introducción de estas bases de conocimiento distingue las "InteRRaP" de las "Touring Machines".

La forma en que las diferentes capas en "InteRRaP" funcionan para producir el comportamiento también difiere de las "Touring Machines". La principal diferencia es la forma en que las capas interactúan con el entorno. En las "Toruing Machines" cada

capa tenía como entrada las percepciones del entorno y como salida una acción en el entorno. Esto introducía la necesidad de un módulo de control para tratar los conflictos. En "InteRRaP", las capas interactúan con el resto para conseguir la misma finalidad. Los dos tipos de interacción entre capas son la activación "bottom-up" y la ejecución "top-down". La activación "bottom-up" ocurre cuando una capa inferior pasa el control a una capa superior porque no es competente para tratar con la situación actual. La ejecución "top-down" ocurre cuando una capa superior hace uso de los recursos que provee una capa inferior para conseguir sus objetivos. El flujo básico de control comienza cuando una entrada (percepción) llega a la capa más baja de la arquitectura. Si la capa de comportamiento puede tratar con esta entrada, entonces lo hará. En caso contrario, ocurrirá una activación hacia arriba y el control será pasado a la capa de planificación. Si la capa de planificación puede manejar la situación, entonces lo hará, normalmente haciendo uso de la ejecución hacia abajo. En otro caso, usará una activación hacia arriba para pasar el control a la capa más alta. De esta forma, el control fluirá de la capa más baja a las capas más altas de la arquitectura y posteriormente en sentido descendente otra vez.

Bibliografía

[Brenner,1998] Walter Brenner, Rüdiger Zarnekow y Hartmut Wittig, *Intelligent Software Agents. Foundations and Applications*, Ed. Springer-Verlag, 1998.

[Finin,1994] Tim Finin, Don McKay, Rich Fritzson y Robin McEntire, "KQML: An Information and Knowledge Exchange Protocol", en Kazuhiro Fuchi y Toshio Yokoi (eds.), *Knowledge Building and Knowledge Sharing*, Ohmsha and IOS Press, 1994.

[Gómez,1999] A. Gómez-Pérez y V. R. Benjamins, "Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods", *Proc. of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods* (KRR5), 1999, Stockholm, Sweden.

[Hayes,1985] B. Hayes-Roth, "A Blackboard Arquitectures for Control", *Artificial Intelligence*, 1985, vol. 26, number 3, pág. 251-321.

[Jagannathan,1989] V. Jagannathan and R. Dodhiawala and L. S. Baum (ed.), *Blackboard Architectures and Applications*, Academic Press, 1989.

[Labrou,1997] Yannis Labrou y Tim Finin, *A Proposal for a new KQML Specification*, TR CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, February 1997.

[Llorens,2001] Faraón Llorens, *Sistemas de Razonamiento y Conocimiento Distribuido. Agentes Inteligentes*, Tesis Doctoral, Dpto. de Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante, 2001.

[Neches,1991] R. Neches, R. E. Fikes, T. Finin, T. R. Gruber, R. Patil, T. Senator y W. R. Swartout, "Enabling Technology for Knowledge Sharing", *AI Magazine*, 1991, vol. 12, number 3, pág. 36-56.

[Nii,1986a] H. P. Nii, "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Arquitectures", AI Magazine, 1986, vol. 7, number 2, pág. 38-64.

[Nii,1986b] H. P. Nii, "Blackboard Systems (Part Two): Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective", *AI Magazine*, 1986, vol. 7, number 3, pág. 82-106.

[Nilsson,2001] Nils J. Nilsson, *Inteligencia Artificial. Una nueva síntesis*, Ed. McGraw-Hill, 2001.

[Odell,2001] James J. Odell, H. Van Dyke Parunak y Bernhard Bauer, "Representing Agent Interaction Protocols in UML" (pág. 121-140), en *Agent-Oriented Software Engineering*, Paolo Ciancarini y Michael Wooldridge eds., Springer-Verlag, 2001.

[Omicini,2001] A. Omicini, F. Zambonelli, M. Klusch y R. Tolksdorf (eds.), *Coordination of Internet Agents. Models, Technologies, and Applications*, Springer-Verlag, 2001.

[Rosenschein,1994] J. S. Rosenschein y G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, MIT Press, 1994.

[Searle,1969] John R. Searle, *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, 1969. (ed. Cast.: *Actos de Habla*, Cátedra, 1986)

[Searle,2001] John R. Searle, *Mente, lenguaje y Sociedad. La Filosofía en el Mundo Real*, Alianza editorial, 2001.

[Url_AUML] *AUML: Agent UML*, http://www.auml.org.

[Url_FIPA] *FIPA: The Foundation for Intelligent Physical Agents*, http://www.fipa.org .

[Url_KQML] *Knowledge Query and Manipulation Language*, http://www.cs.umbc.edu/kqml.

[Weiss,2000] Gerhard Weiss (ed.), *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, 2000.