xx(8611.1)

CINVESTAV
Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

LCIASA: Lenguaje de "Capacidad de Interacción" de Agentes en Sistemas Abiertos

Tesis que Presenta
Nicandro Farías Mendoza

Para obtener el grado de
Maestro en Ciencias

En la especialidad de
Ingeniería Eléctrica

Guadalajara, Jal. Septiembre de 2000

LCIASA: Lenguaje de "Capacidad de Interacción" de Agentes en Sistemas Abiertos

Tesis de Maestría en Ciencias
Ingeniería Eléctrica
Por:

Nicandro Farías Mendoza

Licenciado en Física y Matemáticas
Instituto Politécnico Nacional 1978-1983

Director de Tesis:

Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Septiembre de 2000

Agradecimientos

Agradezco a mi esposa

Lorena Carmina Moreno Jiménez Por su comprensión y por su apoyo vigoroso.

A mis hijos:

Katia Carmina

Víctor Uriel

Isaac

Por quienes hago este esfuerzo.

Agradecimientos:

A mi asesor

Dr. Félix F Ramos Corchado
Por su ayuda, por su confianza, por sus enseñanzas y por su infinita paciencia y comprensión

A los miembros del comité de tesis

Dr. Félix F. Ramos Corchado
Dr. Luis Ernesto López Mellado
Dr. Víctor Hugo Zaldívar Carrillo

A la Universidad de Colima. Por brindarme su patrocinio para realizar mis estudios
Al Instituto Tecnológico de Colima. Por brindarme las facilidades para continuar superándome académicamente

A todos mis compañeros de generación por su gran apoyo

Indice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1 Definición del problema

Los sistemas de procesamiento de información contemporáneos, demandan cambios que les permitan adecuarse a las necesidades actuales de manejo de la información.

En nuestros días los sectores industriales, comerciales, gubernamentales, educativos y en general todos los sectores productivos y sociales tienen la necesidad apremiante de integrarse en un contexto de información global, en este se tienen diferentes tipos de aplicaciones y diferentes plataformas de operación, todo en un ambiente trasparente al usuario.

De lo anterior surgen necesidades de integración, organización y operación, tanto de la información como de los dispositivos físicos que conforman los sistemas de información, esto hace indispensable el desarrollo de herramientas tecnológicas, con las cuales se pueda hacer frente los requerimientos de información actual y en el futuro.

Por lo expuesto en los párrafos precedentes y por la información obtenida en el estado del arte de los sistemas multiagente (como se expone posteriormente en el capítulo 2), se identifica que el problema central esta dado porque los lenguajes para agentes de software (KQML, JATLite, JAFMAS, OpenDoc, etc.), carecen de un modelo formal, con el cual sea posible verificar la equivalencia, consistencia y veracidad de los sistemas especificados por estos lenguajes, en contextos de operación distribuida abierta.

1.2 Motivación

Para resolver el problema mencionado anteriormente se propone el lenguaje LCIASA. El Lenguaje LCIASA que se propone contiene un Metalenguaje de agentes denominado AML (por sus siglas en inglés), el cual está asociado, a LCIASA, como se verá posteriormente en el capítulo 4.

Con LCIASA, es posible hacer una especificación formal de un sistema mutiagente, utilizando el metalenguaje AML. Esta especificación formal la utilizamos para modelar el sistema especificado. El modelo obtenido es utilizado como una herramienta de diseño de sistemas multiagente, es decir con este modelo es posible conocer el comportamiento, alcances y propiedades del sistema especificado, antes de instrumentarlo como un sistema computacional.

Por otra parte LCIASA, que corresponde al lenguaje objeto de agentes AOL (por sus siglas en inglés), es avistado como un dialecto de KQML [28]. Es decir LCIASA está soportado por la estructura y funcionalidad de los mensajes de KQML, con LCIASA se describe la interacción (intercambio de mensajes o diálogo que sostiene los agentes para llegar a un acuerdo) entre los agentes.

LCIASA ofrece las siguientes ventajas referente a los lenguajes de agentes de software y en particular sobre KQML:

- simplicidad, claridad y mayor expresividad en los mensajes que lo conforman.
- Formalización de las especificaciones de sistemas multiagente por medio del metalenguaje AML.
- Aplicaciones robustas en sistemas distribuidos abiertos.

LCIASA representa una herramienta apropiada para desarrollar aplicaciones en ambientes distribuidos, así mismo con LCIASA es posible adecuarse a los nuevos modos de procesamiento de la información (Comercio Electrónico, Realidad Virtual, etc.), estos y otros aspectos como la flexibilidad a los cambios de contexto en las diversas aplicaciones de software, resaltan la importancia de LCIASA.

LCIASA es considerado como un lenguaje de sexta generación por las siguientes razones: primera, LCIASA ofrece facilidades para producir aplicaciones distribuidas robustas en contextos abiertos. Segunda porque

LCIASA tiene un metalenguaje formal asociado con el cual es posible crear un modelo para estudiar las propiedades, similitudes y alcance de los sistemas especificados. Tercera, LCIASA es un lenguaje de alto nivel, declarativo, clasificado como un lenguaje de comunicación humana, por su orientación hacia expresiones en lenguaje natural.

A continuación se mencionan otros aspectos que motivan el desarrollo de este trabajo de tesis:

La tecnología del procesamiento de la información ha sufrido cambios importantes en las últimas décadas. Los sistemas de información han pasado de ser manuales a ser automatizados, asimismo los sistemas centralizados están emigrando hacia sistemas distribuidos.

Los avances tecnológicos en el hardware, software y medios de comunicación, así como los avances en la investigación teórico-práctica asociados a la computación, permiten una comunicación más apropiada a nuestra sociedad (personas, empresas, etc.).

Muchos de los sistemas de información actuales operan bajo el poco eficiente esquema centralizado el cual requiere grandes recursos tanto de software como de hardware para administrarlo. Sin embargo con los avances de las tecnologías mencionadas, los sistemas de información están emigrando hacia los esquemas de operación de información distribuida. Como ejemplos de dicha evolución se citan los siguientes:

i) Las herramientas de programación de la Internet proporcionan ambientes realmente distribuidos, los cuales conforman la base sobre la cual los sistemas multiagente pueden desarrollarse.

ii) Asimismo al surgir herramientas de software como los lenguajes de programación(OO, visuales, etc.), los manejadores de bases de datos Orientados a Objetos, las plataformas multiagentes (CORBA) y los lenguajes y protocolos de comunicación entre agentes (KQML, KIFF, OLE, etc.), han favorecido el desarrollo de sistemas distribuidos.

Los agentes móviles representan una nueva tecnología que está alimentando a una industria renovada. Como consecuencia de que tanto la tecnología como la industria son nuevas, los agentes móviles difieren en su arquitectura y en su implementación.

Estas diferencias entre los sistemas de agentes móviles entorpece la interoperabilidad y la rápida proliferación de la tecnología de agentes y tal vez impida el crecimiento de la industria.

Para promover tanto la interoperabilidad como el desarrollo de diversos sistemas multiagente se deben estandarizar aspectos como: transferencia de

agentes, nombres de agentes y sistemas de agentes, tipos de sistemas de agentes y sintaxis de localización.
En un futuro se vislumbran grandes sistemas de procesamiento de información basados en conocimiento, compuestos de muchos organismos dinámicos en interacción.
Del mismo modo se tendrán aplicaciones de agentes que recorran las redes de información realizando tareas específicas.
Una de las ventajas de los sistemas multiagentes es su potencial para desarrollar un rango amplio de aplicaciones en ambientes distribuidos.
Los Sistemas MultiAgente (MAS por sus siglas en ingles), particularmente las redes de agentes autónomos con comportamiento no impuesto ofrecen enfrentar mejor los retos de la demanda de agilidad en la industria de manufactura, control y calendarización.
Las industrias cuyas aplicaciones son inherentes a las redes verán el desarrollo más rápido en la tecnología de los MAS.
Como ejemplo se citan las aplicaciones en la industria, en donde se comprueba que los grandes sistemas de software centralizados no son tan efectivos como las redes computacionales de agentes distribuidos(GE Power Generation Jpb Shop, Archon, The Flavors Paint Shop, Hitachi ADS, etc.). [15]
Asimismo se manifiesta la importancia de los MAS por su amplio rango de campos de aplicación en donde son empleados, entre otros se mencionan los siguientes:

Solución de problemas cooperativos y AI distribuida.

La IAD considera los aspectos de cómo un grupo de agentes pueden cooperar con el fin de resolver eficientemente los problemas y como las actividades de tal grupo pueden ser coordinadas eficientemente, los estudiosos de IAD han aplicado la tecnología de agentes en varias áreas, ejemplos de aplicaciones incluyen: sistemas de potencia, control de tráfico aéreo, recuperación inteligente de documentos, redes de telecomunicaciones, ingeniería concurrente, planificación laboral etc.

Agentes de Interfaz.

Un agente de interfaz es un agente que actúa como una clase de asistente inteligente a un usuario respecto a una aplicación computacional, mucho del trabajo sobre agentes de interfaz es hecho por la comunidad de trabajo cooperativo por computadora (CSCW)

Agentes de información y sistemas de información cooperativo.

Un agente de información es un agente que ha accesado una y potencialmente muchas fuentes de información y es capaz de recolectar y manipular la

información obtenida de estas fuentes y responder a preguntas de los usuarios o de otros agentes de información se tienen dos prototipos de agentes de información: IRA (Information Retrieval Agents) que es capaz de buscar un artículo que haya sido vagamente especificado, en un conjunto de documentos y CARNOT que permite que un sistema de bases de datos heterogéneo pueda integrarse para responder preguntas que están fuera del alcance de cualquier BD individual.

Agentes creíbles.

Son agentes que proporcionan la ilusión de vida, tal agente tiene aplicaciones potenciales en juegos por computadora y ambientes de cinema virtual, en la actualidad se están desarrollando arquitecturas de agentes creíbles. (grupo OZ de CMU) [15].

1.3 Objetivos

El objetivo general del trabajo de tesis consiste en desarrollar un lenguaje para lograr la interacción entre agentes, el cual consiste de un conjunto de acciones colectivas para lograr un propósito o meta establecida.

El lenguaje está basado en el paradigma "Capacidad de Interacción", que describe un modelo basado en planes para intercambiar información entre Agentes, el lenguaje consiste de un conjunto de sentencias expresadas como una locución cotidiana, de manera similar al lenguaje que utilizan los humanos para intercambiar información.

Este paradigma esta definido por dos componentes básicas: La filosofía del paradigma y el soporte del paradigma con los cuales se describen completamente los principios que lo caracterizan.

Los objetivos específicos del trabajo de tesis son:

- Proponer una metodología para el estudio de la Interacción entre agentes, en donde éstos agentes están inmersos en un sistema distribuido abierto. La metodología propuesta abarca: La definición y aspectos teóricos de los conceptos de agentes y sistemas multiagentes (cap. 2), el modelo formal que describe los sistemas multiagente (cap. 3), el paradigma "Capacidad de Interacción" que genera al lenguaje LCIASA (cap. 4) la arquitectura del agente (cap. 4), y el lenguaje LCIASA para definir la interacción entre los agentes (cap. 4).

El concepto Interacción, que se utiliza en este trabajo de tesis tiene la siguiente acepción: Intercambio de acciones significativas entre miembros de una sociedad, comunicación lingüística, en la destaca el papel igualmente activo que desempeñan los interlocutores al construir con sus respectivos enunciados un universo de discurso conversacional con sus implicaciones sociales, su trascendencia sobre la visión del mundo y las creencias, etc.

- Desarrollar un lenguaje formal, basado en la lógica temporal de creencias, para la especificación de los sistemas multiagente, denominado Meta Lenguaje de Agentes (AML por sus siglas en inglés). Éste lenguaje está asociado al lenguaje LCIASA y es utilizado como un lenguaje de especificación formal de Sistemas MultiAgente.
- Implementar un lenguaje basado en el paradigma "Capacidad de Interacción", para describir la interacción entre los agentes, denominado Lenguaje de "Capacidad de Interacción" de agentes en Sistemas Abiertos (LCIASA).
- Implementar un soporte para el desarrollo de sistemas basados en conocimiento.
- Crear una herramienta que facilite implementar aplicaciones de comunicación móvil.
- Desarrollar tecnología de Software cooperativo.

1.4 Estructura de la tesis

La tesis está organizada en cinco capítulos, en ellos se presenta un panorama que muestra una idea completa de la interacción en Sistemas MultiAgente, desde sus conceptos básicos, evolución, fundamentos teóricos hasta su especificación.

El capítulo 2 presenta las definiciones y conceptos de agente y Sistemas MultiAgente, también se presenta una evolución de los conceptos básicos de los agentes como son: La teoría de Agentes, su arquitectura, los lenguajes de comunicación para agentes y sus aplicaciones.

En el capítulo 3 se proponen los formalismos para especificar tanto a los agentes como la interacción entre los mismos, indicando sus características,

reglas sintácticas y reglas semánticas. Estos formalismos representan un apoyo para modelar Sistemas MutiAgente, con esto es posible verificar, conocer las propiedades y alcances de los sistemas en estudio.

El capítulo 4 muestra el paradigma "Capacidad de Interacción", y la arquitectura del agente. Éstos representan el soporte para originar el lenguaje de interacción entre agentes LCIASA y como consecuencia los protocolos de interacción asociados al lenguaje.

En el capítulo 5 se presenta un resumen de los resultados obtenidos, expresando los comentarios pertinentes al desarrollo de este trabajo, asimismo se dan las conclusiones y se hacen recomendaciones para continuar este trabajo en un futuro.

Capítulo 2

Agentes y Sistemas MultiAgente

Objetivo: en Éste capítulo se presentan las definiciones y conceptos de agente y Sistemas multiAgente, también se muestra una evolución de los conceptos básicos de los agentes como son: sus conceptos teóricos, su arquitectura y los lenguajes de comunicación para agentes. En este capítulo proporciona la base con la cual se define la metodología para concretar el lenguaje propuesto.

El concepto de agente ha tomado una importancia significativa tanto en la inteligencia artificial (IA) como en las ciencias de la computación. Para remarcar su importancia consideremos que una manera de definir la inteligencia artificial es diciendo que es un subcampo de la ciencia de la computación, la cual permite construir agentes que exhiben aspectos de comportamiento inteligente. De tal forma que el concepto de agente es central al campo de la IA. Generalmente el concepto de agente es utilizado para denotar hardware o más comúnmente sistemas de computación basados en software.

2.1 Definiciones y conceptos

Realmente no existe una definición estándar de agente para la cual se tenga un consenso, los investigadores de esta área han propuesto varias definiciones para este término y aún lo siguen haciendo debido a que una definición refleja una percepción particular del término. Sin embargo, con el fin de precisar el concepto de agente se presentan las siguientes definiciones:

FIPA (Foundation for Intelligent Physical Agents) define un agente como:

Una entidad que reside en medios ambientes en donde percibe (por medio de sensores) los datos producidos por eventos en el medio ambiente y ejecuta comandos que producen efectos en el medio ambiente. Un agente puede ser puramente software o hardware, en el último caso se requiere una cantidad considerable de software para convertir un hardware en un agente.

Los agentes también son conocidos con los siguientes nombres:

- Demonios(ejemplo un agente ftp)
- Interfaces usuario clientes (agente de correo)
- Agentes físicos (robótica)
- Agentes creíbles (realidad virtual y gráficas)
- Agentes de software inteligente

Pattie Maes[19] del MIT da la siguiente definición:
Un agente es un sistema computacional que habita en un medio ambiente dinámico complejo. El agente puede percibe y actuar en su medio ambiente y tiene un conjunto de metas y motivaciones que trata de alcanzar por medio de sus acciones.

Considerando su aplicación agente puede ser: [7]

- Un recuperador de Información basado en agentes
- Agentes de pantalla de usuario
- Agentes inteligentes de interfaz hombre-máquina
- Agentes de modelado adaptativo para el usuario
- Asistente personal (experto)
- Tecnología de software móvil
- Agentes de software cooperativo
- Agentes para la investigación
- Mediadores y facilitadores
- Agentes de mercado electrónico

2.1.1 Propiedades de los agentes

Autonomía. Se dice que un agente es autónomo, si éste puede operar sin la intervención directa de humanos o de sus usuarios y si tiene alguna clase de control sobre sus acciones y su estado interno. Se puede considerar que un agente es autónomo en la medida que su conducta esté definida por su propia experiencia.

Reactividad. Los agentes tienen la habilidad de percibir su medio ambiente, que puede ser el mundo físico o un usuario por medio de una interfaz o una colección de otros agentes y responder a los cambios que ocurran en este.
Habilidad social. Los agentes establecen una interacción con otros agentes y posiblemente con humanos, utilizando alguna clase de lenguaje de comunicación entre agentes
Cooperación. Para llevar a cabo sus tareas, generalmente un agente es considerado como un sistema de cómputo que además de las características anteriores, se le asocian conceptos que usualmente son aplicados a los humanos, como el conocimiento, las creencias, intenciones, obligación y aún más la emocionalidad. La cooperación entre agentes es un mecanismo por el cual los agentes intercambian su conocimiento, sus creencias y sus planes con el fin de trabajar juntos y resolver problemas complejos, los cuales estarían mas allá de sus capacidades individuales.
Pro-actividad. Los agentes no solamente actúan en respuesta a su medio ambiente sino que estos son capaces de exhibir un comportamiento propio, tomando la iniciativa. Éstos pueden razonar acerca de sus intenciones y creencias y tomar las acciones correspondientes.
Racionalidad. Un agente actúa para alcanzar sus metas y no considera las posibles contingencias al tratar de alcanzar sus objetivos.
Varias otras propiedades son asociadas a los agentes como la *movilidad* que es la habilidad de un agente para desplazarse a través de una red electrónica. La *veracidad* con la que se asume que un agente no comunica información falsa y la *adaptabilidad* que permite que un agente se adecue continuamente a los cambios de su medio ambiente.

2.1.2 Clasificación de los agentes

Con las propiedades consideradas anteriormente para los agentes, los agentes se pueden clasificar de la siguiente forma:
Agentes autónomos son aquellos que habitan en un medio ambiente dinámico, complejo al cual percibe y entonces actúa de manera independiente en este medio ambiente, realizando una serie de tareas para alcanzar una meta.
Agentes de información son agentes que tienen acceso a muchas fuentes de información y tienen la habilidad de manipular dicha información para responder las preguntas propuestas por el usuario o por otros agentes. Algunas veces se les conoce como agentes de Internet.

Agentes inteligentes. Son los agentes que llevan acabo un conjunto de operaciones de interés para un usuario o para otro agente, con algún grado de independencia.

Agentes de interfaz. Esencialmente los agentes de interfaz dan soporte y proporcionan asistencia al usuario. El agente observa y percibe las acciones realizadas por el usuario en la interfaz, aprende de estas acciones y sugiere una manera mas apropiada de realizar la tarea, como se muestra en la fig. 4.1. Los agentes de interfaz generalmente aprenden a proporcionar una mejor asistencia de las siguientes cuatro formas:

- Por la observación e imitación del usuario
- A través de la retroalimentación positiva o negativa del usuario
- Recibiendo instrucciones explícitas del usuario
- Preguntando por consejos a otros agentes

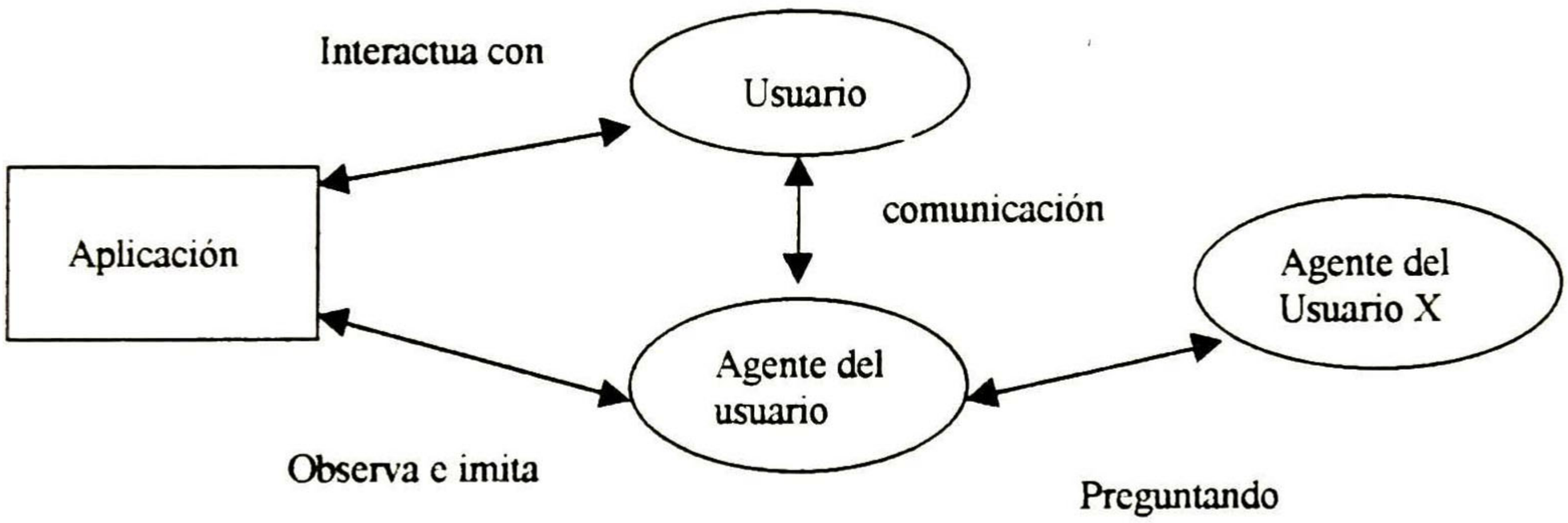


Fig. 2.1 Esquema de un agente de interfaz

Agentes de entretenimiento. Éstos agentes se utilizan con el propósito de entretenimiento como juegos, producciones de vídeo.

Agentes colaborativos. Con estos agentes se da mayor importancia al la autonomía y cooperación con respecto a otros agentes con el fin de ejecutar sus tareas propias. Los atributos principales de estos agentes incluyen la autonomía, habilidad social, responsabilidad y la pro-actividad. Con el fin de alcanzar una inicialización coordinada los agentes requieren intercambiar información (negociar) con el fin de tener un acuerdo aceptable, al realizar una tarea conjunta.

Agentes móviles. Són procesos con la capacidad de transitar por una red que puede ser local o de cobertura amplia, tal como Internet, interactuando con anfitriones remotos para obtener la información requerida por el usuario o

bien para ejecutar una tarea que le ayude a alcanzar su objetivo y retornar a su lugar de origen.
Con el atributo de movilidad se introduce el concepto de programación remota, en donde los agentes envían, además de datos, el procedimiento que se va a ejecutar, logrando una actuación como pares, en donde cada agente puede proceder ya sea como cliente o como servidor. [2]
Bajo este esquema se tiene una reducción significativa del tráfico de la red. Marcando una diferencia significativa con la programación cliente-servidor. Como se ilustra en la fig. 2.2 [2]

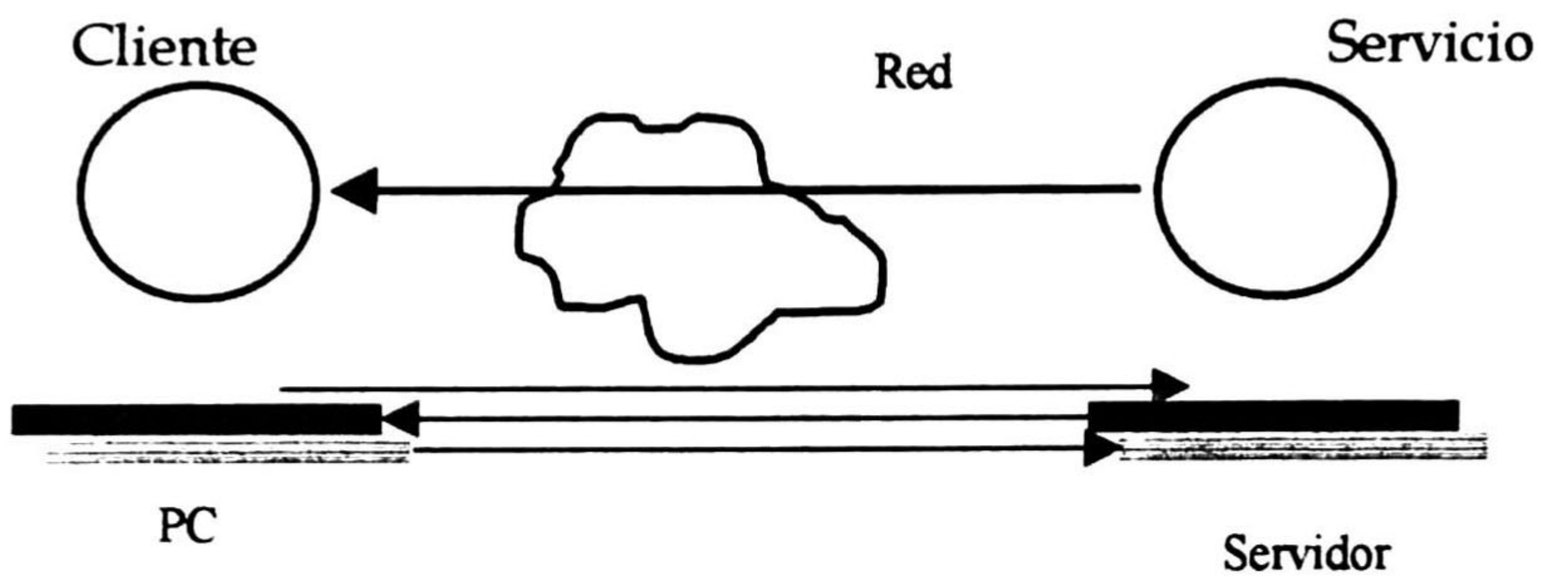


Fig.2.2 Programación Cliente-servidor y Programación Remota

Algunos de los puntos más importantes a considerar, en la implementación de agentes móviles son: la seguridad, los dispositivos para mantener la información secreta, los mecanismos de transporte y autentificación. Además los sistemas deben ser protegidos contra virus, ciclos infinitos en el CPU y otros riesgos. A continuación se mencionan algunos de los principales agentes móviles.

AgentTcl de la Universidad de Dartmouth
Telescript de General Magic
Odyssey de General Magic Inc. IBM
Aglets Workbench IBM Japon
Voyager de Objectspace Inc.
Concordia de Mitsubishi Electric

Agentes Reactivos. Los agentes reactivos representan una categoría especial los cuales no poseen modelos simbólicos internos, ellos actúan y responden de la forma estimulo-respuesta para presentar estados del medio ambiente que los rodea.
Éstos agentes están basados en los siguientes tres principios básicos [20]:

- La dinámica de interacción entre estos agentes no inteligentes debe conducir a una complejidad emergente.
- Un agente reactivo generalmente es visto como una colección de módulos, los cuales operan de manera autónoma y son responsables de tareas específicas (sensar, percibe, cálculos, etc.). Las comunicaciones entre estos módulos son mínimas y de bajo nivel
- Un agente reactivo tiende a operar sobre las representaciones que están cerca del sensor de datos, en contraste con las representaciones de alto nivel utilizadas por otro tipo de agentes.

Frecuentemente los agentes reactivos son llamados NMAS (Natural Multi-agent System). Los NMAS son colonias de agentes que muestran un comportamiento emergente y cooperativo para proporcionar un comportamiento integrado del sistema, asumiendo que los agentes individuales son inteligentes.
Agentes híbridos. Como se observó en los agentes mencionados anteriormente, se tienen ciertas deficiencias para cada uno de ellos, lo que hace necesario maximizar las cualidades y minimizar las deficiencias, Pattie Maes [20] propuso adoptar una aproximación híbrida, que tuviera las cualidades de los paradigmas deliberativos y reactivos.
La conformación de un agente híbrido combina dos o más definiciones de comportamiento de agentes en un solo agente. Los agentes híbridos pueden presentar características móviles, de interfaz, colaborativas etc. Una implementación interesante de esta paroximación híbrida es la arquitectura de agente InteRRap desarrollada por (DFKI).

2.1.3 Sistemas MultiAgente

Los Sistemas multiAgente son un desarrollo de la comunidad de inteligencia artificial distribuida. Un Sistema MuliAgente (MAS por sus siglas en ingles) se define como: la colección de agentes que pueden comunicarse por medio de *una red débilmente acoplada. Los agentes trabajan en conjunto para resolver problemas que están mas allá de las capacidades individuales de un agente.*
Los agentes pueden ser por su condición heterogénea, son caracterizados por varios grados de capacidad para resolver problemas, generalmente tienen un único contexto de control y/o intención.
Los investigadores de Sistemas multiAgente están principalmente interesados en la coordinación del comportamiento inteligente entre estos agentes, en como éstos coordinan su conocimiento, metas, técnicas y planes en forma conjunta y así tomar las acciones conducentes a la solución de problemas.

2.1.3.1 Consideraciones de implementación

Para obtener un comportamiento coherente del sistema, los agentes individuales en un Sistema MultiAgente, deben no solo ser capaces de compartir conocimiento acerca del problema y soluciones, deben también razonar acerca del proceso de coordinación entre otros agentes. La tarea de coordinación puede ser muy difícil de lograr en un Sistema MultiAgente. Las dificultades inherentes encontradas en la implementación de un comportamiento coordinado en cualquier Sistema MultiAgente fueron identificadas por Gasser[12], estas son las siguientes:

i). *Comunicación*: definir cómo hacer posible que los agentes se enlacen y que protocolo utilizar.
ii). *Interacción*: Especificar que lenguaje han de utilizar los agentes para interactuar con cada uno de los otros y poder combinar sus esfuerzos para resolver un problema específico.
iii) *Coherencia y coordinación*: se trata de asegurar que los agentes se coordinan con cada uno de los otros para obtener una solución coherente al problema que se está tratando de resolver.

2.1.3.2 Comunicación

La comunicación permite que los agentes en un Sistema MultiAgente puedan intercambiar información basándose la estructura de la organización adaptada por el SMA para alcanzar su objetivo. Es decir en un Sistema MultiAgente se debe contar con varios caminos para que los agentes intercambien información entre ellos. Los agentes pueden *intercambiar directamente mensajes* o pueden organizarse ellos mismos en un *sistema federado* y comunicarse a través de agentes facilitadores especiales o pueden *difundir el mensaje* (broadcast).

Otra aproximación popular que permite a los agentes comunicarse entre si es por *medio de un pizarrón* (blackboard) en donde la información pueda ser depositada y recuperada por los diferentes agentes.

Comunicación directa: la comunicación directa implica el establecimiento de enlaces físicos directos con otros agentes utilizando algún protocolo, como TCP/IP el cual asegura una entrega confiable de paquetes de mensajes. Los enlaces físicos implican que el agente debe estar consciente de los otros agentes en el sistema. Las direcciones de los agentes pueden ser obtenidas ya sea por los mensajes recibidos en la difusión por otros agentes o por un objeto centralizado, por ejemplo el AgentNameServer en JATlite, que es como un servicio de directorio donde todos los agentes conforman el sistema de registro. Un agente transmisor puede tener acceso a la dirección del receptor observando este objeto centralizado.

La especificación de FIPA de 1997 especifica que todo Sistema MultiAgente debe tener un directorio de agentes, el cual contiene información de todos los agentes en un medio ambiente particular y facilita la identificación y el acceso de agentes. Un mecanismo de comunicación directa es utilizado en muchos de los lenguajes y plataformas de construcción de agentes. La comunicación directa tiene sentido cuando un agente sostiene un diálogo y este sabe exactamente a quien enviar el mensaje.

Sistemas federados. [1] Cuando el número de agentes en un sistema se hace muy grande, el costo y el procesamiento requerido en la comunicación directa es prohibitivo. Una alternativa para la comunicación directa que elimina estas dificultades consiste en organizar el conjunto de agentes en un sistema federado.

Como se observa en la figura 2.3 los agentes no se comunican directamente ente ellos, sino que se comunican a través de un *facilitador* (mediador) de agentes especial. Cada facilitador (interfaz común de los agentes) posee información de las necesidades y habilidades de un subgrupo de agentes.

Bajo este esquema los agentes pueden enviar solicitudes e información al facilitador, así como aceptar las aplicaciones y solicitudes del mismo. El facilitador emplea la información del agente para transformar estos mensajes y enrrutarlos a los sitios apropiados.

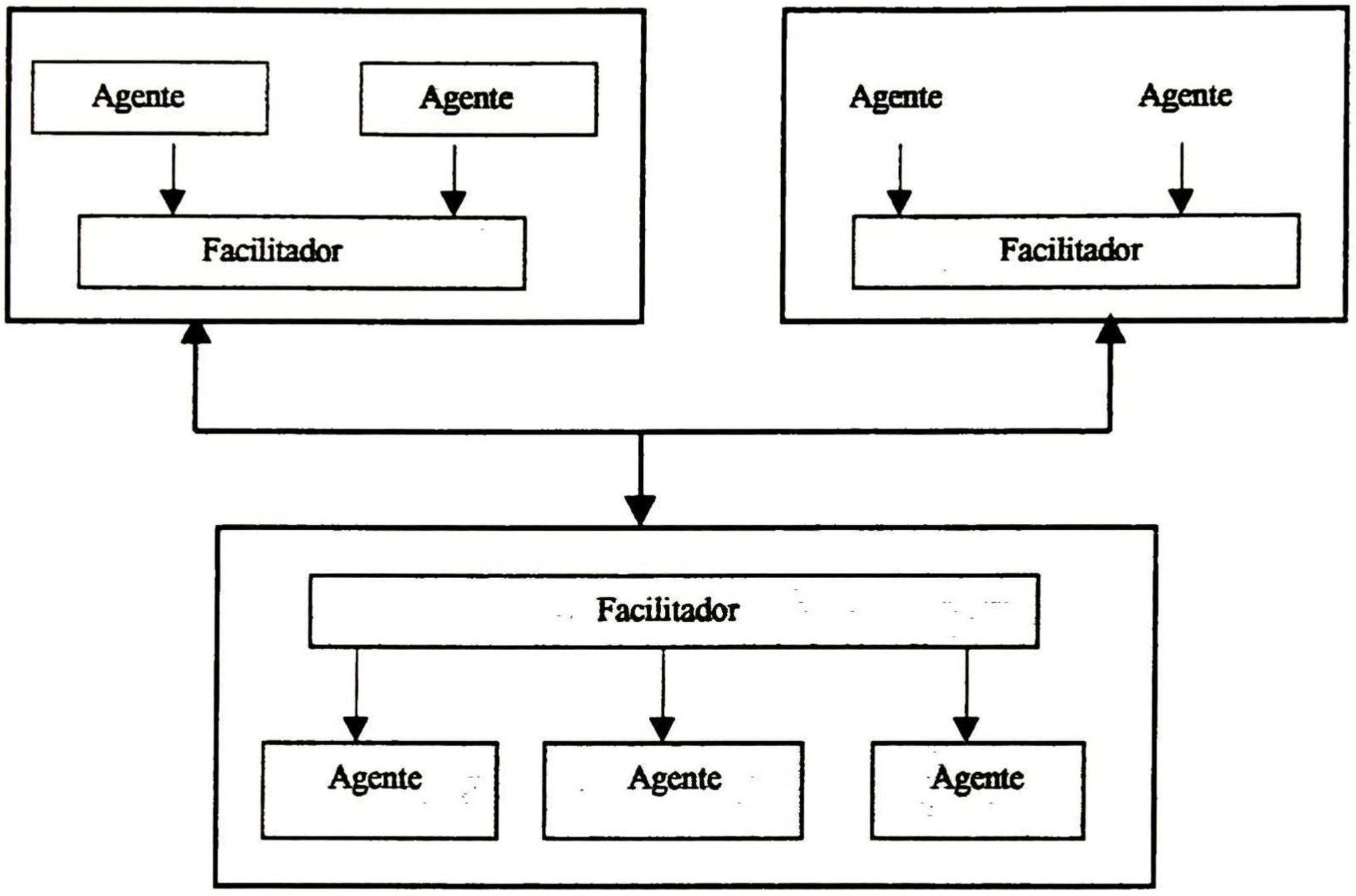


Figura 2.3: Sistema federado

Comunicación por difusión. En situaciones en que un agente tenga que comunicarse con todos los agentes de su medio ambiente, o cuando el transmisor no sabe quien será el receptor, se tienen dos opciones: se puede difundir físicamente el mensaje a todos los agentes del sistema o puede mantener enlaces de comunicación individual con todos los agentes del sistema y enviar a cada uno de ellos un mensaje directo. Esta última opción tiene problemas cuando el numero de agentes es muy grande debido a que absorbe un ancho de banda considerable. La comunicación por difusión a través del "web" es conocida como "webcastig".

Sistemas de pizarrón. En IA el pizarrón es empleado como un modelo de comunicación. Éste es un depósito en donde los agentes escriben mensajes,

reparten resultados parciales que son útiles a otros agentes quienes obtienen información de este repositorio. La información es particionada en varios niveles de abstracción, de acuerdo al problema tratado. Los agentes que están trabajando en un nivel de abstracción particular tienen acceso a su correspondiente nivel del pizarrón así como a los niveles adyacentes. Mientras que las metas de los niveles mas altos pueden filtrarse hacia abajo para manejar las expectativas de los agentes de los niveles más bajos.

2.1.3.3 Interacción: Teoría del acto de conversación

El argumento segundo en importancia para la implementación en un MAS es la interacción que significa: un tipo de acciones colectivas de donde un agente toma una acción o hace una decisión que ha sido influenciada por la presencia o conocimiento de otro agente. Este es inherentemente un concepto distribuido ya que esta basado en las acciones coordinadas de los agentes participantes. Debido a que las acciones en el sistema son usualmente dirigidas a metas. Muchas interacciones son derivadas de las metas.

La naturaleza heterogénea y distribuida de un MAS, provoca que la implementación de interacción entre agentes sea un proceso dificultoso. Entonces cuando se trabaje con agentes distribuidos es importante diseñar un lenguaje común expresivo para la comunicación con un agente independiente de la semántica. en donde los agentes puedan comunicarse con sus semejantes por medio del intercambio de mensajes e interactuar conjuntamente empleando acciones lingüísticas explícitas.

En realidad esto es en donde los agentes difieren de los objetos, los objetos pueden interactuar con otros accesando métodos públicos dependientes del objeto, estos métodos pueden diferir de un objeto a otro. El lenguaje de comunicación en un MAS debe ser independiente de los agentes. Los agentes deben comunicarse a través de una interfaz de mensajes común, que es independiente de su estructura de datos interna. Esto conduce a la necesidad de saber que conocimiento representar para la comunicación y como hacerlo. Como los agentes comunicantes tendrán diferentes bases de conocimiento, el lenguaje debe admitir estas diferencias con el fin de que pueda darse la comunicación y la cooperación a pesar de estas discrepancias. Entonces cada agente debe tener una capa lingüística para soportar una semántica independiente del agente.

En la comunidad de MAS la teoría del acto de conversación(speech-act) es uno de los métodos más comunes para construir la capa lingüística y formalizar las acciones lingüísticas de los agentes. La teoría del acto de conversación ha

hecho contribuciones importantes al entendimiento de las relaciones entre un estado interno de un agente y los enunciados que intercambia con otros agentes. Ésta teoría usa el concepto de tipos de mensajes para permitir a un agente transmitir sus creencias, deseos e intenciones. Los tipos de mensajes son los componentes del acto de conversación del lenguaje y determina que se puede hacer o ejecutar con el contenido del mensaje. Por ejemplo los tipos de mensaje: "assert", "afirm", "state", transmiten una creencia, los tipos de mensaje: "ask", "order", "enjoin", "pray" o "command", transmiten un deseo y los tipos de mensaje: "vow", pledge", "promise", transmiten una intención.
Un lenguaje de acto de conversación que es utilizado habitualmente en la comunidad de multiagentes es el KQML. Este facilita la cooperación de alto nivel y la interoperación entre agentes artificiales. Aquí los agentes pueden variar entre programas simples y bases de datos hasta sistemas basados en conocimiento más sofisticados y estos se comunican pasando tipos de mensajes a cada uno de los otros, los tipos de mensajes de KQML forman una parte importante del lenguaje. KQML soporta muchos tipos de mensajes diferentes.

2.1.3.4 Coherencia y coordinación

Este tema consiste en asegurar la coherencia y coordinación del sistema global. Gasser [12] diferencia la coherencia y coordinación refiriéndose a la coherencia para analizar que tan bien se comporta el sistema completo cuando está resolviendo un problema y examinando el comportamiento conjunto del sistema y a la coordinación como la propiedad de interacción entre un conjunto de agentes ejecutando una acción colectiva. Jennings[18] afirma que sin coordinación el beneficio de los sistemas descentralizados se desvanece y la comunidad puede rápidamente deslizarse hacia una colección de individualidades caóticas e incoherentes. Los dos términos están relacionados en el sentido que a una mayor coordinación resulta una solución más coherente al problema integral. Las tres razones principales propuestas por Jennings[18] para indicar porqué las acciones de múltiples agentes deben ser coordinadas son las siguientes:

- Existen dependencias entre las acciones de los agentes, las cuales pueden ocurrir cuando las metas individuales de los agentes están relacionadas.
- Hay una necesidad de encontrar restricciones globales. Estas existen cuando la solución que se está desarrollando por el sistema debe satisfacer ciertas condiciones si va ser considerada exitosa.

- Los agentes individuales no tienen la suficiente aptitud, recursos o información para resolver el problema cabal. Muchos problemas no pueden ser resueltos por agentes trabajando incomunicado porque no poseen la suficiente experiencia, recursos o información.

Si Todos los agentes del sistema pudieran tener un conocimiento completo de las metas, acciones, e interacciones de sus miembros semejantes en la comunidad y pudieran tener también una potencia infinita de procesamiento, sería entonces posible conocer exactamente que estuvo haciendo un agente en el presente y que tratará de hacer en el futuro. En tales condiciones sería posible evitar los conflictos y los esfuerzos redundantes y el sistema sería perfectamente coordinado. Pero debido a las limitaciones en los recursos físicos como el ancho de banda, la memoria etc., éste sistema es irrealizable.
Se han propuesto varias aproximaciones acerca de la coherencia y coordinación, [1] [3] algunos autores proponen que el problema de la coordinación en un MAS debe ser estudiado a un nivel organizacional. Sugieren que el problema puede ser atacado, teniendo conocimiento acerca de los procesos de interacción que se están llevando a cabo entre agentes, en esta aproximación a la coordinación de agentes, generalmente se forman planes, los cuales especifican sus acciones e interacciones futuras respecto de un objetivo en particular que se trate de alcanzar. Parunak[23] introduce un formalismo alternativo: la gráfica de Dooly[9] para representar la interacción de agentes. Cada nodo de la gráfica de Dooly en una conversación, representa un participante en una etapa distinguida del un discurso, en donde una etapa distinguida esta definida en términos de resolución y completud. La conversación que resuelve o completa a otra tiende a formar componentes fuertemente conexas de la gráfica, mientras que aquellas que toman nuevas direcciones expanden nuevas componentes como veremos posteriormente en el capítulo 4.

2.1.3.5 Ventajas de los multiagentes

Los sistemas mutiagentes proporcionan un medio para desahogar las restricciones del control secuencial centralizado, para proporcionar sistemas que son descentralizados, emergentes y concurrentes. La importancia de los sistemas multiagentes es debida a que muchas de las aplicaciones son inherentemente distribuidas, algunas son espacialmente distribuidas, otras son funcionalmente distribuidas. Las ventajas obtenidas al utilizar sistemas basados en agentes autónomos son las siguientes:

Coordinación autónoma. La ventaja principal de los sistemas multiagente es la propiedad que tienen los agentes de coordinarse de manera autónoma para alcanzar un objetivo, que ningún agente podría alcanzar de manera individual.

Tolerancia a fallas. Como los agentes son mecanismos por naturaleza distribuidos, entonces un sistema elaborado con agentes autónomos no se colapsa cuando uno o más de sus componentes falla.

Software modular/arquitectura escalable. Una razón de que los agentes sean entidades poderosas es debido a la modularidad de un problema que estos proporcionan, esto es debido a que cada agente es visto como una entidad, lo que ayuda a obtener un crecimiento gradual y una expansión flexible. La ventaja de la escalabilidad es proporcionada en la medida que cada agente: pueda formar un sistema, pueda trabajar con otros agentes o pueda dejar el sistema una vez que haya terminado su tarea.

Sistemas autoconfigurables. Una población de agentes puede reconfigurarse así mismos cuando están corriendo (en ejecución). Esta es una ventaja importante para los sistemas que deben responder a un rango amplio de condiciones diferentes.

Reducción del costo de Software. En la medida en que el software sea más modular, el tiempo de desarrollo y su complejidad se reducen, por lo que el costo del software decrece.

Reducción de los costos del Hardware. En la implementación de sistemas basados en agentes se requiere usar CPUs más baratas, lo que disminuye los costos del hardware.

Solución de problemas más rápida. Debido a que en un MAS podemos explotar de manera natural el paralelismo, esto conduce a una ágil solución de problemas.

Sistemas Flexibles. Se logra al tener agentes con diferentes habilidades trabajando en forma dinámica para resolver problemas.

2.2 Teoría de agentes

Una teoría de agentes es considerada como una especificación para un agente, los investigadores estudiosos de la teoría de los agentes, [20] desarrollaron formalismos para representar las propiedades de los agentes y utilizan estos formalismos para desarrollar teorías que capturen las características deseadas de los agentes.

2.2.1 Conceptos de la teoría de agentes

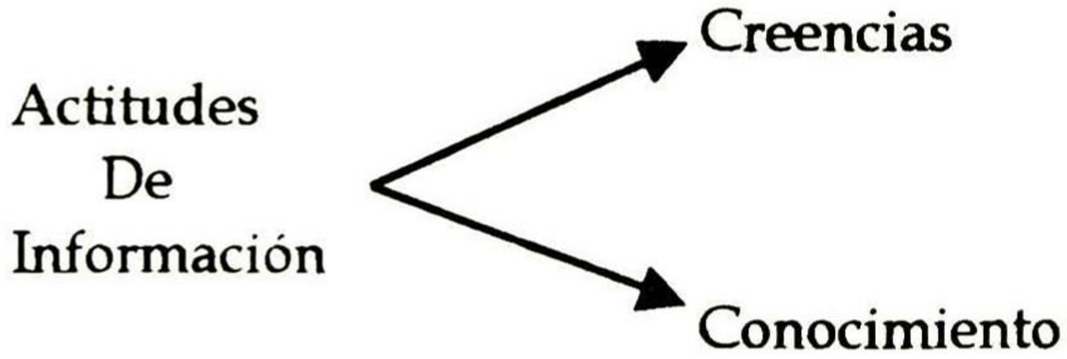
El punto de partida es la noción de agente, como una entidad, la cual parece estar sujeta a creencias, deseos etc. El filósofo Dennet designó el término: *Sistema intencional* para denotar a los agentes y para describir entidades cuyo comportamiento se puede predecir atribuyendo creencias, deseos y clarividencia racional.

Una teoría de agentes completa debe definir la forma en que los componentes de un agente están relacionados, es decir debe mostrar como la información de los agentes y las pro-actitudes están relacionadas, asimismo debe precisar como un estado de conocimiento de un agente cambia con el tiempo, como el medio ambiente afecta a un estado de conocimiento del agente y como la información y las pro-actitudes de un agente lo llevan a ejecutar acciones. El hecho de proporcionar resultados satisfacibles de estas relaciones, es el problema más significativo al cual se dedican los investigadores de la teoría de los agentes.

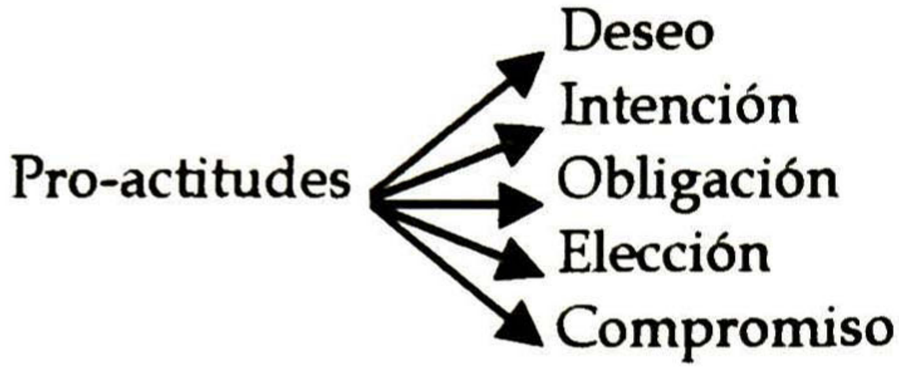
Se pretende que una teoría de agentes realista estará representada en una estructura lógica que combine estos aspectos. Adicionalmente esperamos que una lógica de agentes sea capaz de representar los aspectos dinámicos de los agentes.

2.2.2 Actitudes para la representación de agentes

Las actitudes de información están relacionadas a la información que un agente tiene acerca del mundo que este ocupa:



Por otra parte las pro-actitudes son aquellas que de alguna manera guían las acciones de los agentes



2.2.3 Representación de nociones intencionales

Hay dos aproximaciones fundamentales para la representación sintáctica y semántica de las nociones intencionales, para el problema de la *sintaxis* se tiene el lenguaje de la *lógica modal*, la cual contiene operadores modales, que son apropiados para la formulación de las pro-actitudes. Una aproximación alternativa implica el uso de un *meta-lenguaje* : un lenguaje multi-ordenado de primer orden, el cual contiene términos que denotan la formulación de algún otro lenguaje-objeto. Las nociones intencionales pueden representarse adecuadamente usando predicados en metalenguaje.

Para el problema de la semántica se tienen también dos aproximaciones: *La primera* es la semántica de los *mundos posibles* en donde las creencias, conocimientos metas, etc. de un agente son caracterizados como un conjunto de los llamados mundos posibles, con una relación de accesibilidad entre ellos. La semántica de los mundos posibles tiene asociada una teoría de correspondencia, que la hacen una herramienta matemática atractiva para trabajar con ella.

La segunda alternativa para el modelo de los mundos posibles, consiste en usar una *aproximación sentencial* o estructura simbólica interpretada. En este esquema, las creencias son vistas como una formulación simbólica, representada explícitamente en una estructura de datos asociada con el agente.

2.2.4 Semántica de mundos posibles

El modelo de los mundos posibles para la lógica del conocimiento y creencias fue originalmente propuesto por Hintikka y ahora es comúnmente formulado en una lógica modal normal, utilizando las técnicas desarrolladas

por Kripe. La idea de Hintikka consistió en visualizar las creencias de un agente, caracterizándolo como un conjunto de mundos posibles.
Desafortunadamente la formulación modal normal de creencias y conocimientos tiene serias desventajas, la principal es la omnisciencia lógica, que consiste en que tal lógica predice que los agentes creen todas las consecuencias lógicas de sus creencias. Claramente esta predicción no es razonable para cualquier agente con recursos limitados, pero todos los sistemas reales son de recursos limitados, por los tanto los agentes generalmente creen solo algunas de las consecuencias lógicas de sus creencias, es decir los agentes no creen todas las consecuencias posiblemente lógicas.

2.2.5 Alternativa al modelo de mundos posibles

Como resultado de las dificultades con la omnisciencia lógica muchos investigadores han tratado de desarrollar formalismos alternativos para representar creencias. A continuación se mencionan dos de las propuestas más expresivas:
Levesque,. creencia y conciencia En 1984 Levesque propuso una solución para el problema de la omnisciencia lógica, que consistió en hacer una distinción entre creencias implícitas y explícitas. La idea consiste en asumir que un agente tiene relativamente un número pequeño de creencias explícitas y muchos (infinito) conjuntos de creencias implícitas, las cuales incluyen las consecuencias lógicas de las creencias explícitas. El modelo de Levesque fue sujeto a varias objeciones. Como un esfuerzo para recuperar estos aspectos negativos *Fagin y Halpern*, desarrollaron una lógica de conciencia general, basada en la idea de Levesque pero con una semántica más simple.
Konolige, el modelo de deducción. El modelo de deducción de creencias es en esencia un intento directo para modelar las creencias de sistemas simbólicos de IA, Konolige observó que un sistema típico basado en conocimiento tiene dos componentes básicas : *una base de datos de creencias* representada simbólicamente que pueden tomar la forma de reglas, marcos, redes semánticas etc. Y de algún *mecanismo de inferencia*, lógicamente incompleto. Este sistema es simple y es recomendado como un modelo directo para los sistemas de creencias en IA.

2.2.6 Evolución de la teoría de agentes.

Ahora se presenta una revisión breve de las teorías de agentes más significativas.

Conocimiento y acción. Moore fue en muchos sentidos un pionero del uso de la lógica para capturar aspectos de los agentes, su interés principal fue el estudio de pre-condiciones de conocimiento para acciones. A la interrogante ¿qué necesita un agente conocer para llevar a cabo una acción? . Él formalizó un modelo de habilidad en una lógica conteniendo una modalidad para el conocimiento y un mecanismo de lógica-como (logic-like) dinámico, para modelar las acciones. Este formalismo permitió la posibilidad de que un agente tenga la información incompleta acerca de cómo alcanzar una meta y de la ejecución de acciones con el fin de encontrar la manera de alcanzar dicha meta.

Intención. Cohen y Levesque [4] hicieron una de las contribuciones con mayor influencia en el área de la teoría de agentes, su formalismo fue utilizado originalmente para desarrollar una teoría de intenciones, que los autores requirieron como prerequisito para una teoría de actos de expresión, en donde la lógica probó ser útil para el razonamiento de agentes, la cual fue utilizada en un análisis de conflictos y cooperación en un diálogo multi-agente, así como en varios estudios de los fundamentos teóricos de resolución de problemas cooperativos.

Arquitecturas de intenciones, deseos y creencias . Rao y Georgeff [25], Como se observó anteriormente no existe consenso claro, ya sea en la IA o en las comunidades filosóficas acerca de precisar cual combinación de información y pro-actitudes son las más adecuadas para la caracterización de agentes racionales. Rao y Georgeff, [25] desarrollaron una estructura lógica para la teoría de agentes, basada en tres modalidades primitivas: creencias, deseos e intenciones. Su formalismo está basado en un modelo de ramificación del tiempo, en el cual los mundos accesibles de creencias, deseos e intenciones, son estructuras de ramificación del tiempo.

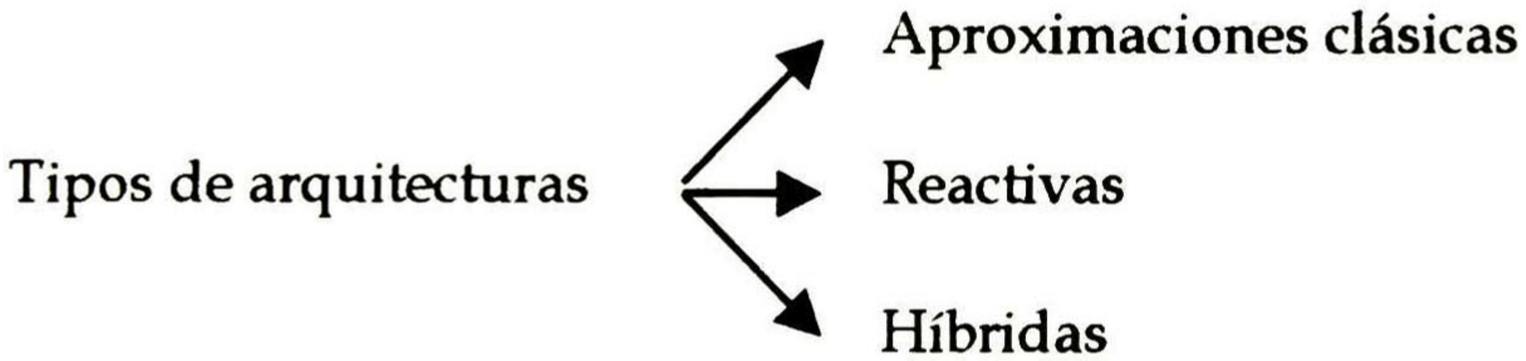
Familia de lógicas. Una aproximación diferente para modelado de agentes fue considerada por Singh, quien ha desarrollado una familia de lógicas , interesante para representar intenciones, creencias, conocimiento, know-how y comunicación en una estructura de ramificación del tiempo. El formalismo de Singh es muy rico y se ha dedicado un esfuerzo considerable para establecer sus propiedades.

Modelo general. En una secuencia extensa de artículos, Werner [31] propone la arquitectura para la cual ha establecido los fundamentos de un modelo general de agentes, con aplicación en economía, teoría de juegos, teoría de autómatas situados, semántica y filosofía. Aunque las propiedades del modelo no han sido estudiadas aún en profundidad.

Modelado de sistemas multi-agente. Para su tésis doctoral Wooldridge desarrollo una familia de lógicas, para representar las propiedades de los sistemas multi-agente. El objetivo de Wooldridge [32] no fue desarrollar una estructura general para la teoría de agentes, mas bien intentó construir formalismos que pudieran usarse en la especificación y verificación de sistemas multi-agente , para realizarlos desarrollo un modelo simple y en algún sentido genérico de sistemas multi-agente y mostró como las historias rastreadas en la ejecución de tales sistemas podría utilizarse como un fundamento semántico para una familia de lógicas lineales y lógicas de creencias temporales de ramificación del tiempo. Él dió ejemplos de cómo estas lógicas podrían utilizarse para la especificación y verificación de protocolos para acciones cooperativas.

2.3 Arquitectura de agentes

La noción básica de la Arquitectura de agentes consiste en definir los mecanismos que permitan pasar de teoría a la práctica, es decir, es decir se consideran los tópicos que posibiliten la construcción de sistemas computacionales que satisfagan las especificaciones dadas por los modelos teóricos. [7]



2.3.1 Aproximaciones clásicas

La aproximación clásica para la construcción de agentes, consiste en visualizarlos como un tipo particular de sistema basado en conocimiento, este paradigma es conocido como *inteligencia artificial simbólica.*

Un tipo de arquitectura clásica es la llamada *arquitectura deliberativa* que son aquellas que contienen un modelo simbólico explícito del mundo, en el cual las decisiones son hechas por medio de un razonamiento lógico, basado en un empate de patrones y la manipulación simbólica, La fig. 2.4 muestra un esquema de éste tipo de arquitectura.

Las aproximaciones clásicas son teóricamente atractivas, aunque parece muy difícil de llevar a la práctica, el problema subyacente consiste en la dificultad de probar los teoremas, aún en lógicas muy simples, ya que estas tienden a ser irresolubles.

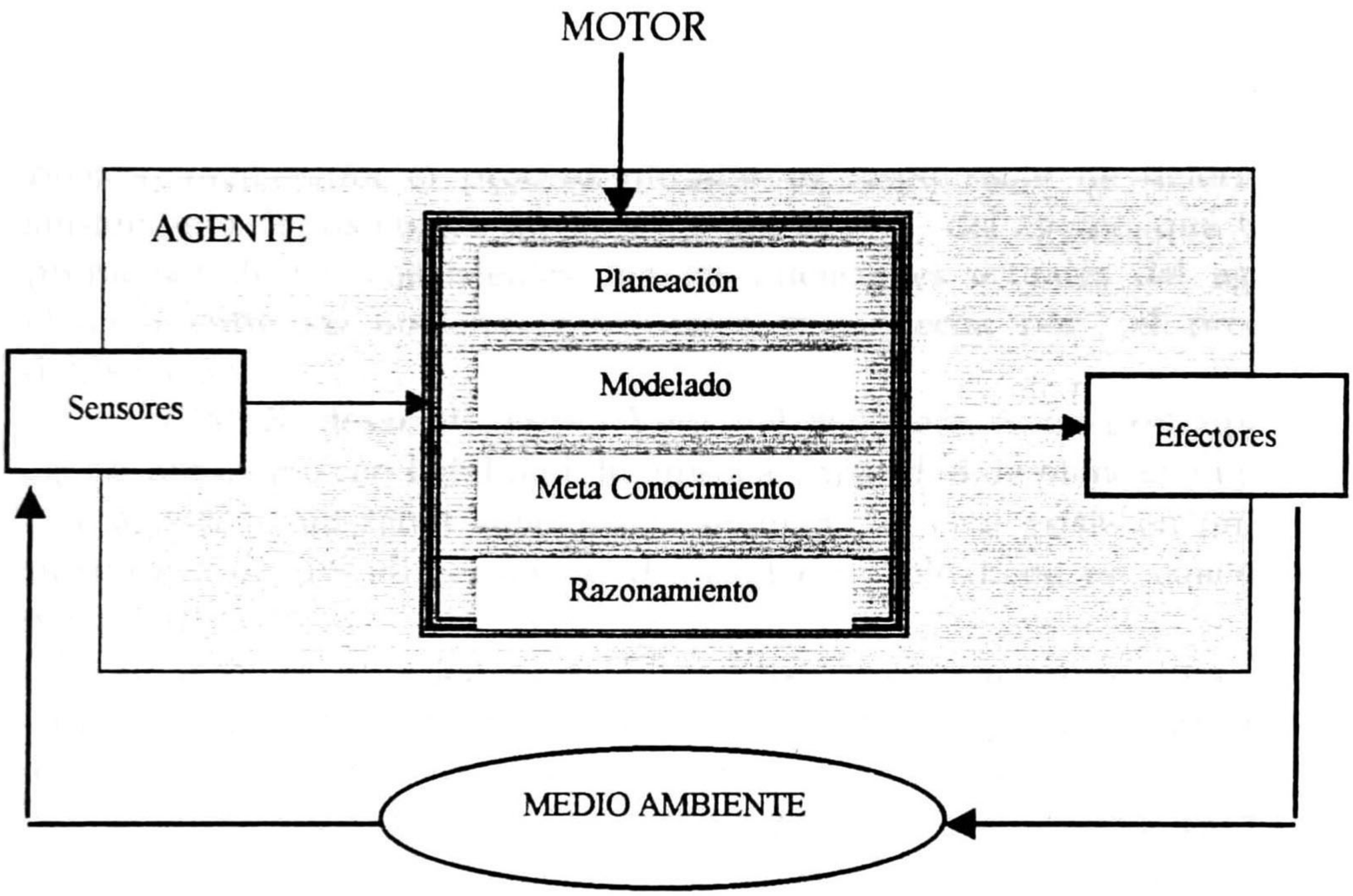


Fig. 2.4: Arquitectura clásica

Otra arquitectura clásica es la de *agentes de planeación*, en donde la planeación es esencialmente la programación automática, es decir el diseño de un curso de acción que cuando se ejecuta permite alcanzar los objetivos deseados.

De los primeros sistemas de planeación y posiblemente el más conocido fue STRIPS este sistema toma una descripción simbólica del mundo y el estado de las metas deseadas y un conjunto de descripciones de acciones, las cuales caracterizan las pre-y-post condiciones asociadas con varias acciones. El

algoritmo de planeación de STRIPS fue muy simple y se probó su poca efectividad aún en problemas de moderada complejidad.
A continuación se mencionan las aproximaciones clásicas mas significativas para la arquitectura de agentes .
IRMA (Intelligent Resource-Bounded Machine Architecture) desarrollada por Bratman, Israle and Pollack, esta arquitectura está basada en creencias, deseos e intenciones, contiene cuatro estructuras de datos clave: una *librería del plan*, una representación explícita de creencias, deseos e intenciones. Adicionalmente la arquitectura contiene un *razonador*, que le permite razonar acera del mundo, un analizador de significado terminal, para determinar cuales planes pueden ser usados para alcanzar las intenciones del agente, un *analizador de oportunidades* que percibe el medio ambiente con el fin de determinar opciones posteriores para el agente, un *proceso filtrador* y un *proceso deliberador* el proceso filtrador es responsable de determinar el subconjunto de los cursos de acciones potenciales del agente, que tienen la propiedad de ser consistentes con las intenciones actuales del agente, la elección entre las opciones que compiten es hecha por el proceso de deliberación.

HOMER desarrollado por Vere and Bickmore, es un prototipo de un agente autónomo con habilidad lingüística, capacidad de planeación y acción. Este agente es un robot submarino simulado, el cual existe en un mundo submarino de dos dimensiones, del cual este solo tiene un conocimiento parcial.
GRATE propuesto por Jennings, [17] está basado en una arquitectura de capas en la cual el comportamiento de un agente es guiado por actitudes mentales de creencias, deseos e intenciones e intenciones combinadas.
Los agentes se dividen en dos partes distintas: Un sistema de nivel de dominio y una capa de cooperación y control. La capa de cooperación está compuesta de tres módulos genéricos: un módulo de control el cual proporciona interfaces al sistema de nivel de dominio, un módulo de evaluación de la situación y un módulo de cooperación.

2.3.2 Arquitecturas Reactivas

Como se observo anteriormente, hay muchos problemas insolubles en la inteligencia artificial simbólica, estos problemas condujeron a varios

investigadores a cuestionar la viabilidad del paradigma en su totalidad y a desarrollar lo que se conoce como arquitectura reactiva.
Una arquitectura reactiva es aquella que no incluye ninguna clase de modelo central de mundo simbólico y no utiliza razonamiento simbólico complejo como se representa en la fig. 2.6.
A continuación se mencionan los desarrollos más significativos de arquitecturas de agentes basadas en los principios de la arquitectura reactiva.
PENGI. Agre y Chapman observaron que la actividad diaria mas frecuente es la *rutina* en el sentido de que esta requiere poco, si existe nuevo razonamiento abstracto, muchas tareas una vez aprendidas pueden convertirse en una rutina, con poca variación, Agre y Chapman propusieron que una arquitectura de agente eficiente podría basarse en la idea de argumentos en ejecución, la idea es que como la mayoría de decisiones son rutina, estas pueden codificarse en una estructura de bajo nivel, la cual solo requiere actualizaciones periódicas, tal vez para manejar una nueva clase de problemas.

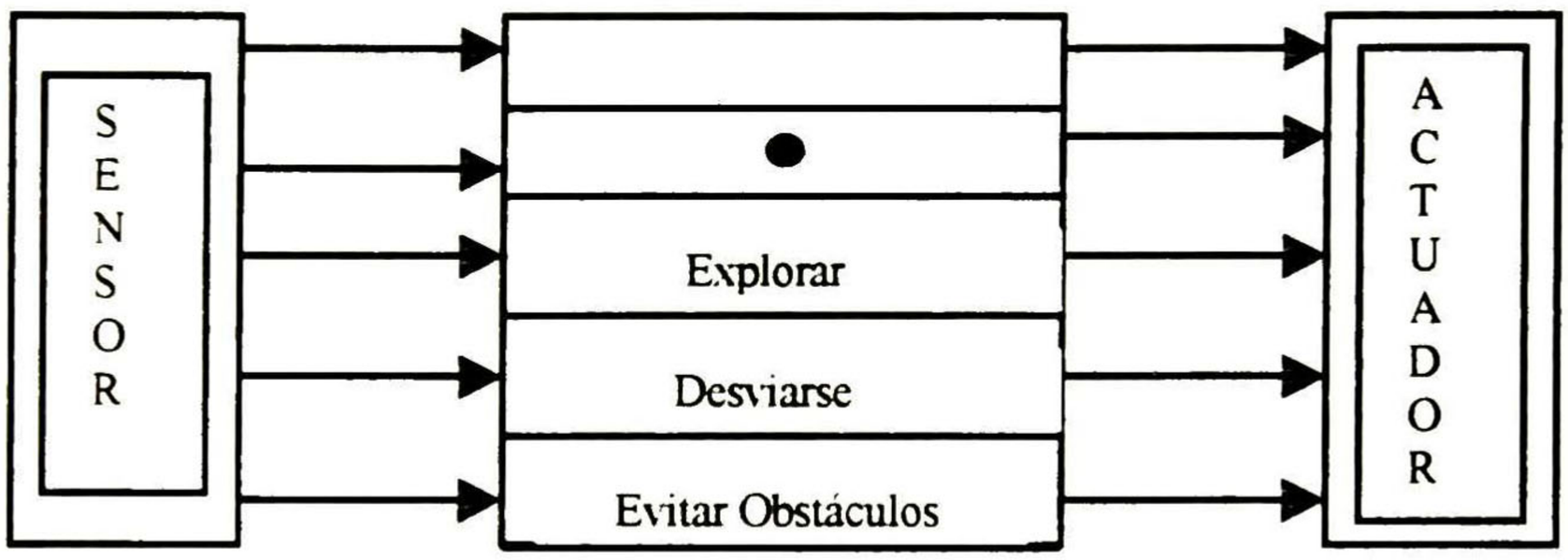


Fig. 2.6: Arquitectura reactiva (Brook´s)

AUTÓMATA SITUADO Otra aproximación sofisticada es la de Rosenschein y Kaelbling. En el paradigma de los autómatas situados, un agente es especificado en términos declarativos, esta especificación es entonces compilada en una computadora digital, la cual satisface la especificación declarativa. El paradigma de autómatas situados ha generado mucho interés, este aparece combinando los mejores elementos de los sistemas reactivos y de los sistemas declarativos
ARQUITECTURA DE RED DE AGENTE Pattie Maes [20] desarrolló una arquitectura de agente, en la cual un agente es definido como un conjunto de módulos de competencia. Existe una similitud entre las arquitecturas de red de agentes y las arquitecturas de redes neuronales. Tal vez la clave de la

diferencia esta en la dificultad de expresar el significado de un nodo en una red neuronal. Este solo tiene significado en el contexto de la red misma. Mientras que los módulos de competencia están definidos en términos declarativos, es mucho más fácil decir cual es su significado.

2.3.3Arquitecturas híbridas

Muchos investigadores han sugerido que ni la aproximación completamente deliberativa ni la completamente reactiva son apropiadas para la construcción de agentes. Se ha considerado el caso de sistemas híbridos que pretenden "cazar" las aproximaciones clásicas y las reactivas. La fig. 2.7 presenta los componentes de este tipo de arquitectura.
A continuación se presentan las propuestas más representativas de este tipo de arquitectura.

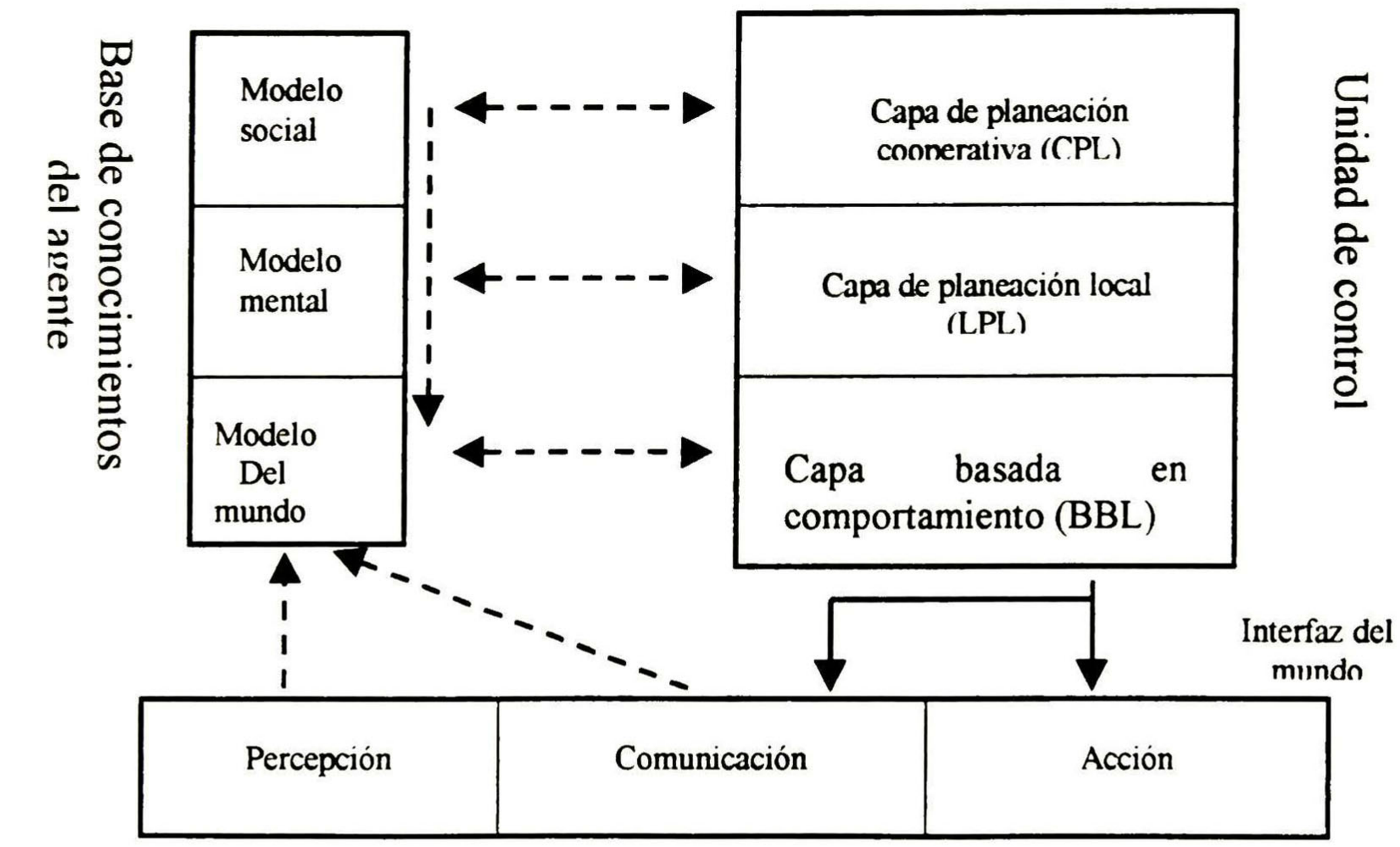


Fig. 2.7: Arquitectura híbrida (InteRRap)

InteRRap: Esta arquitectura fue desarrollada por Muler et al. Esta es una arquitectura en capas en donde cada capa sucesiva representa un nivel mayor

de abstracción que una capa inferior. En InteRRap estas capas están subdivididas en dos capas verticales, una que contiene las capas de las bases de conocimiento y la otra contiene varios componentes de control y el nivel más bajo es la componente de control que proporciona la interfaz con el mundo.

La componente interfaz con el mundo, como su nombre lo indica maneja la interfaz entre el medio ambiente y el agente es decir trata con la percepción, comunicación y la acción. Como lo muestra la fig. 4.7

A continuación se presentan las propuestas más representativas de este tipo de arquitectura.

PRS (Procedural Reasoning System) desarrollada por Georgeff [14] y Lansky, es una de las arquitecturas mas conocidas de agentes, como IRMA, PRS es una arquitectura de creencia-deseo-intención, la cual incluye una librería de planes, las creencias son hechos, ya sea del mundo externo o de los estados internos del sistema y son expresados en la lógica de primer orden, los deseos son representados como comportamientos del sistema. Una Librería de planes en PRS contiene un conjunto de planes parcialmente elaborados, llamados áreas de conocimiento (KAs por sus siglas en ingles) cada una de las cuales está asociada con una condición de innovación, KAs también puede ser reactivo, permitiendo al PRS responder rápidamente a cambios en su medio ambiente. La fig. 2.8 representa éste tipo de arquitectura.

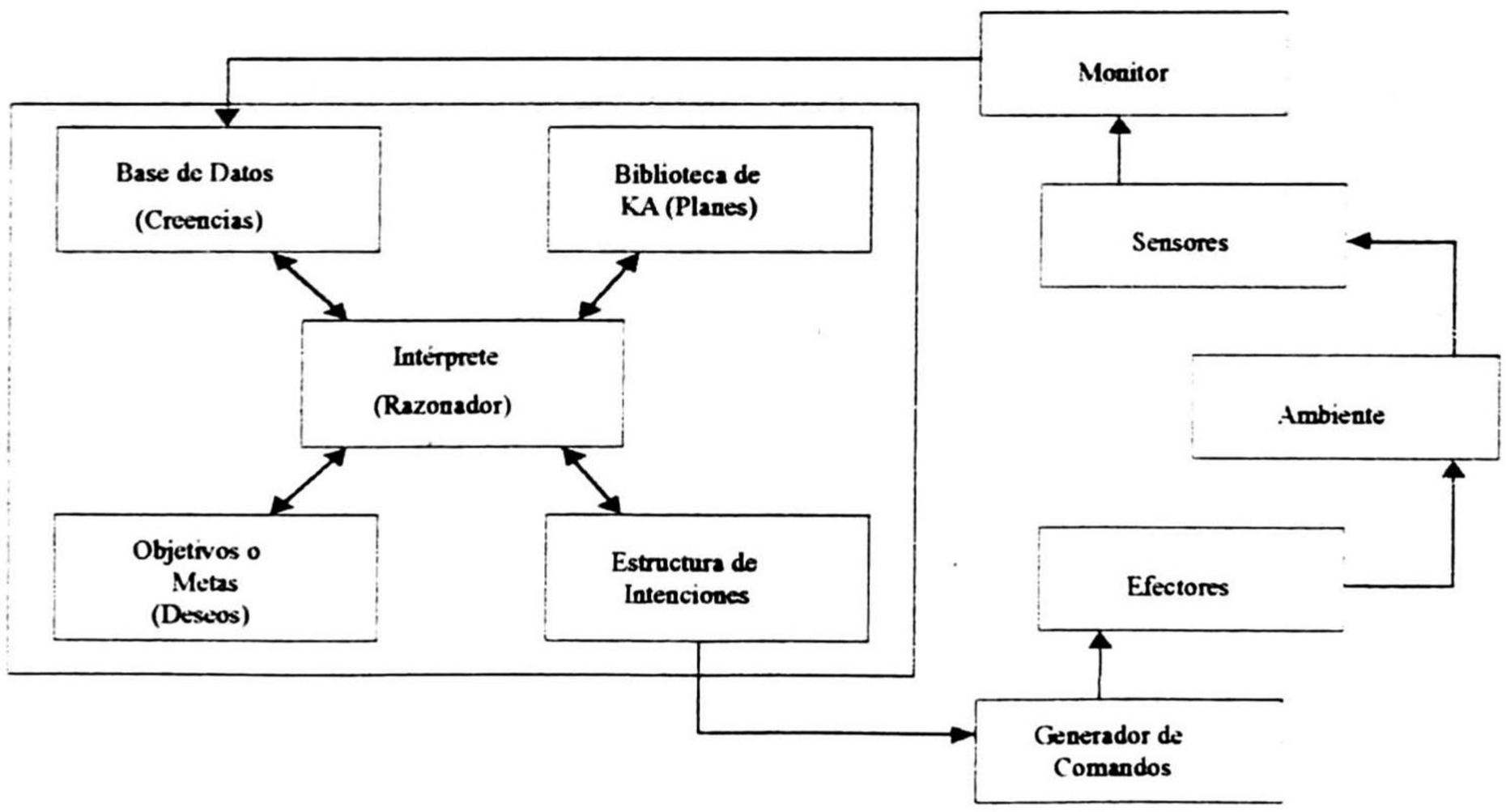


Fig. 2.8 Arquitectura PRS

Otra arquitectura híbrida es COSY la cual combina las características de las anteriores para dar una mayor funcionalidad a la arquitectura de agentes, como lo muestra la fig. 2.9

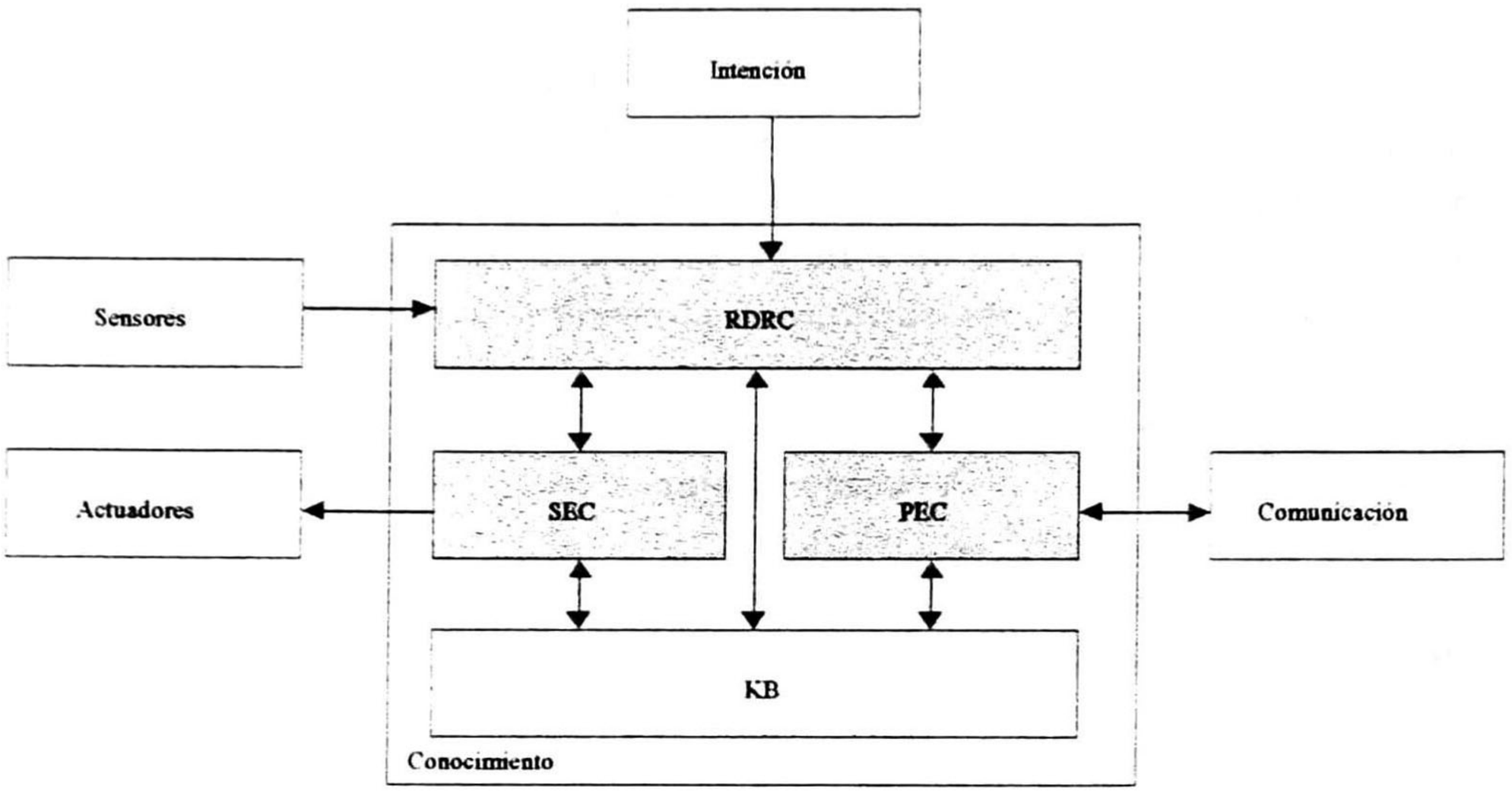


Fig 2.9 Arquitectura COSY

2.4 Lenguajes para comunicación de agentes (LCA)

Los lenguajes de agentes se consideran como sistemas que permiten programar el Software y el Hardware de un sistema computacional en los términos de los conceptos desarrollados por los teóricos para un agente. El lenguaje de agentes debe incluir la estructura que corresponde al agente, de acuerdo a sus propiedades, asimismo podemos visualizar otros atributos de los agentes como: creencias, metas, nociones mentales, etc. Utilizados para programar agentes.

2.4.1 Conceptos de LCA

Los LCA son considerados como la herramienta que utilizan los agentes para comunicarse información y conocimiento. Su importancia está basada

en que el potencial de un SMA se basa en la comunicación entre agentes. Lo cual permite la colaboración y cooperación de los elementos del SMA.

2.4.2 Clasificación de los lenguajes

La categoría de los lenguajes de agentes, abarca a todos los lenguajes que pueden ser usados para implementar agentes de software. En general cualquier lenguaje de programación puede ser utilizado para desarrollar agentes de software. La Tecnología de Agentes de software se divide en dos grandes categorías. Esto se representa a continuación en la fig. 2.10.

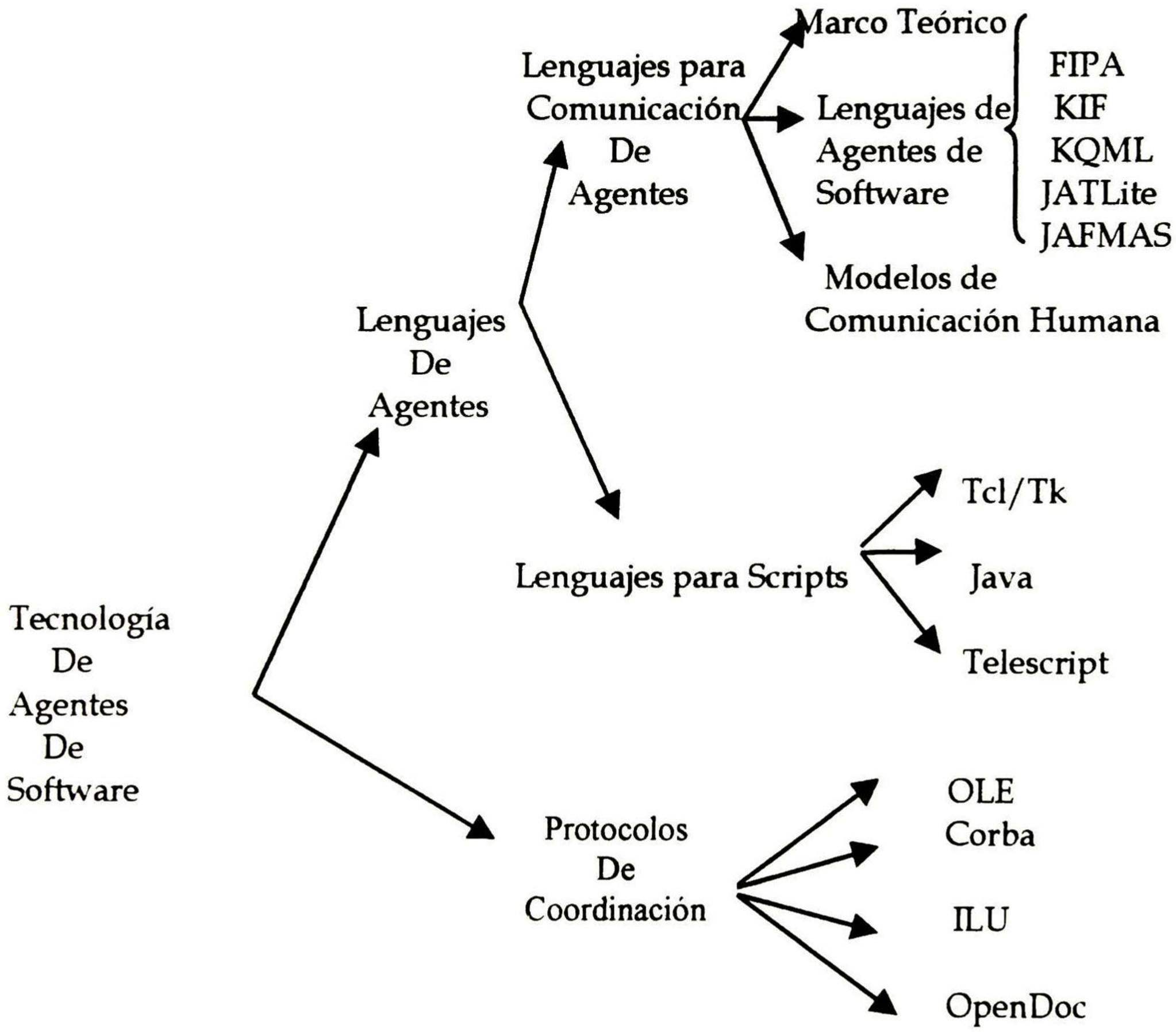


Fig. 2.10 Clasificación de los lenguajes de Agentes

Una clase de lenguajes que ha obtenido mucha atención últimamente son los lenguajes para scripts, especialmente aquellos diseñados para programas móviles, los lenguajes como Java, Telescript, Tcl/Tk etc., ofrecen la ventaja del nivel de abstracción, lo que los hace mas atractivos para el desarrollo de software de agentes. Se pueden diferenciar de los lenguajes de comunicación de agentes (LCA), Porque estos lenguajes fueron diseñados con el propósito principal de controlar procesos en una sola plataforma, es decir están orientados a la transportación de un único agente de una máquina a otra.

En Oposición los lenguajes de comunicación de agentes están diseñados específicamente para describir y facilitar la comunicación entre dos o más agentes. Se tienen tres subcategorías para los LCA: Los modelos de comunicación humana, los marcos teóricos y los lenguajes de comunicación para agentes de software.

La comunicación humana es modelada tradicionalmente en términos de la teoría de actos de expresión (spech acts), de la cual se derivan con frecuencia marcos teóricos para agentes artificiales con capacidades del humano. Estos marcos intentan registrar todos los aspectos de los estados internos de un agente artificial autónomo, tomando atención en particular de los cambios de sus estados cuando el agente sostiene una interacción con el mundo o con otros agentes.

En Contraste los LCA, están interesados estrictamente con la comunicación entre agentes de software. Un LCA es más que un protocolo para intercambiar datos, porque incluso es comunicada una actitud acerca de lo que es intercambiado. Un LCA puede considerarse como un protocolo de comunicación (o colección de protocolos), que soporta muchos tipos de mensajes.

Otra clase vital de la tecnología de agentes de software es aquella que trata acerca de los estándares y de la coordinación de protocolos (CORBA, ILU, OpenDoc, Ole etc.). Los cuales representan esfuerzos que frecuentemente son promulgados como soluciones al problema de la comunicación de agentes. Sin embargo el manejo de estas herramientas implica una dificultad en la ejecución de aplicaciones en ambientes dinámicos distribuidos. El interés principal de estas tecnologías consiste en asegurar que estas aplicaciones puedan intercambiar estructuras de datos y métodos a través de plataformas heterogéneas (disparate).

Aunque los resultados de los esfuerzos de tales estándares sean útiles en el desarrollo de agentes de software, estos no proporcionan respuestas completas al problema de la comunicación entre agentes. Después de todo

los agentes de software son más que una colección de estructuras de datos y métodos en ellos. Se considera que estos estándares y mecanismos de coordinación de protocolos, son más bien vistos como una base sobre la cual, los lenguajes de agentes podrían ser creados.

2.4.3 Evolución de los lenguajes de Agentes

A continuación se menciona los lenguajes más populares para la programación de agentes, los cuales marcaron el inicio a los lenguajes de agentes.

Lenguajes de objetos concurrentes: Los lenguajes de objetos concurrentes son en muchos aspectos los antecesores de los lenguajes de agentes, la noción de auto-contenido, objeto en ejecución concurrente con algunos estados internos que no son accesibles directamente desde el exterior y la respuesta a mensajes de otros objetos, representan una descripción muy cercana al concepto de agente, tal como se ha definido anteriormente. Un sistema muy conocido de objetos concurrentes es el sistema ABCL y el modelo de actor de Hewitt.

Programación orientada a agentes: Yaou Shoham propuso un nuevo paradigma de programación "Agent Oriented Programming" (AOP), con el cual se programan los agentes de manera directa en términos de la mentalística desarrollada por los teóricos, para representar las propiedades de los agentes, tales como conocimiento, creencia, intención etc.

Shoham propuso que un sistema completo AOP debe contener tres componentes principales:

- Un sistema lógico para la definición de los estados mentales de los agentes
- Un lenguaje de programación interpretado para la programación de agentes
- Un proceso de agentificación para la compilación de agentes en sistemas de ejecución de bajo nivel.

Agent 0: Se concibió como un prototipo para ilustrar los principios de AOP, en este lenguaje un agente es especificado en términos de un conjunto de

capacidades (lo que el agente puede hacer), un conjunto de opiniones iniciales, acciones y un conjunto de reglas de acciones. Una implementación más desarrollada, fué instrumentada por Thomas en su lenguaje PLACA (planning Communications Agents).

April y Mail: Son dos lenguajes para desarrollar aplicaciones multi-agentes, April fue desarrollado para proporcionar el núcleo de características necesarias para construir la mayoría de las arquitecturas y sistemas de agentes. La generalidad de April se debe a su capacidad de manejo de abstracciones. El lenguaje Mail proporciona una rica colección de abstracciones predefinidas, incluyendo planes y planes muti-agente, ha sido utilizado para implementar varios prototipos de sistemas multi-agentes.

Telescript: Es un lenguaje basado en ambientes para la construcción de sociedades de agentes, fue desarrollado por General Magic Inc. Es considerado como el primer lenguaje comercial. Hay dos conceptos básicos en la tecnología de Telescript: SITIOS y AGENTES, los sitios son locaciones virtuales que son ocupadas por los agentes, los agentes son proveedores y consumidores de bienes en las aplicaciones de mercado electrónico, desarrolladas por Telescript. Los agentes son procesos de software, son móviles capaces de migrar de un lugar a otro, en tal caso sus programas y estados son codificados y transmitidos a través de una red a otro sitio.
Able: "Agente Behavior Languaje" Fue desarrollado por Connah y Wavish, es un lenguaje en el cual los agentes son programados en términos de reglas simples como-licencias. Las licencias pueden incluir alguna representación de tiempo, este fue desarrollado en lenguaje C.

2.5 Aplicaciones de la Tecnología de Agentes

Solución de problemas cooperativos y AI distribuida: [22] Aunque DAI abarca muchos de los puntos considerados, debe remarcarse que el énfasis clásico de la DAI ha sido sobre los macrofenómenos(nivel social) más bien que en nivel de microfenómenos (nivel de agentes). DAI considera los puntos de cómo un grupo de agentes puede cooperar con el fin de resolver eficientemente los problemas y como las actividades de tal grupo pueden ser coordinadas eficientemente, los estudiosos de DAI han aplicado la tecnología de agentes en varias áreas, ejemplos de aplicaciones incluyen: sistemas de potencia, control

de tráfico aéreo, recuperación inteligente de documentos, redes de telecomunicaciones, ingeniería concurrente, planificación laboral etc.
Agentes de Interfaz: Un agente de interfaz es un agente que actúa como una clase de asistente inteligente a un usuario respecto a una aplicación computacional, mucho del trabajo sobre agentes de interfaz es hecho por la comunidad de trabajo cooperativo por computadora (CSCW)
Agentes de información y sistemas de información cooperativo: Un agente de información es un agente que ha consultado una y potencialmente muchas fuentes de información y es capaz de recolectar y manipular la información obtenida de estas fuentes y responder a preguntas de los usuarios o de otros agentes de información se tienen dos prototipos de agentes de información: IRA (Information Retrieval Agents) que es capaz de buscar un articulo que haya sido vagamente especificado, en un conjunto de documentos y CARNOT que permite que un sistema de bases de datos heterogéneo, pueda integrarse para responder preguntas que están fuera del alcance de cualquier BD individual.
Agentes creíbles: Son agentes que proporcionan la ilusión de vida, tal agente tiene aplicaciones potenciales en juegos por computadora y ambientes de cinema virtual, en la actualidad se están desarrollando arquitecturas de agentes creíbles. (grupo OZ de CMU) [22].

Conclusión: EL contenido del presente capítulo presenta las bases para el estudio de los Sistemas MultiAgente, La estructuración del contenido del capítulo ofrece una metodología para estudiar los conceptos ligados a los SMA.
Este apartado constituye una guía, la cual se tomó como base para la elaboración de este trabajo de tesis, iniciando con los elementos teóricos que sustentan al lenguaje de interacción propuesto, continuado con la arquitectura del agente y finalmente proponiendo el lenguaje que los agentes han de utilizar para intercambiar información y resolver tareas las tareas asignadas.
Los conceptos abordados en este capítulo nos permiten ubicar el lenguaje LCIASA dentro del contexto del estudio de los sistemas multiagente. LCIASA es conceptuado como un lenguaje de comunicación entre agentes (ACL) basado en el modelo de comunicación humana, es un lenguaje de interacción orientado a planes, con los cuales se provee a los agentes de habilidades para realizar una tarea determinada. La arquitectura de agente es del tipo deliberativa, en la que se tiene un modelo simbólico explícito, en donde la decisiones son hechas por medio de un razonamiento lógico basado en la manipulación simbólica.

Capitulo 3
AML: Metalenguaje de Agentes

Objetivo: presentar los formalismos para la descripción de los agentes y Sistemas MultiAgente , asimismo, proponer una LA (lógica de agentes) con su sintaxis y semántica asociada, con la cual se proporcionan los principios para la especificación formal apropiada de Sistemas mutiAgente.
La lógica de agentes LA proporciona un modelo formal para el estudio de los Sistemas MultiAgente, con este modelo es posible ensayar las propiedades y el comportamiento de un sistema de agentes especificado, antes de Instrumentarlo como un sistema computacional.
La lógica temporal utilizada en LA es un tipo de lógica no clásica en donde el modelo del tiempo provee las bases para describir sistemas dinámicos. En este trabajo es utilizada una secuencia discreta lineal en el tiempo de estados, como el modelo temporal básico.

3.1 Comprensión del Proceso de comunicación.

La comunicación entre un objeto (emisor) y un sujeto(receptor) es posible gracias a la naturaleza del entendimiento, el cual consiste en que el sujeto pueda interpretar los símbolos que le fueron enviados por el objeto, relacionándolos con la información almacenada previamente, este análisis es conocido como la interpretación pragmática de un objeto o evento (Prag()).
Supongamos que el receptor tiene una interpretación de su mundo digamos I, al recibir el símbolo Υ el agente receptor tiene que aplicar una interpretación a la información recibida para transformar su percepción del mundo, al obtener la información J.

$$J_{sujeto} = Prag\ (\Upsilon\ )\ I_{sujeto}$$

En donde J es la interpretación que el sujeto le da al símbolo Υ recibido de parte del emisor.
La teoría de la comunicación hace una distinción entre la sintaxis, la semántica y la pragmática de un lenguaje.
Sintaxis: Conjunto de reglas que generan las sentencias gramaticales correctas del lenguaje.

Semántica: relaciona al lenguaje con situaciones o con la representación de situaciones en el mundo, que da sentido a la acción. Esto eso ilustrado en la figura 5.1.

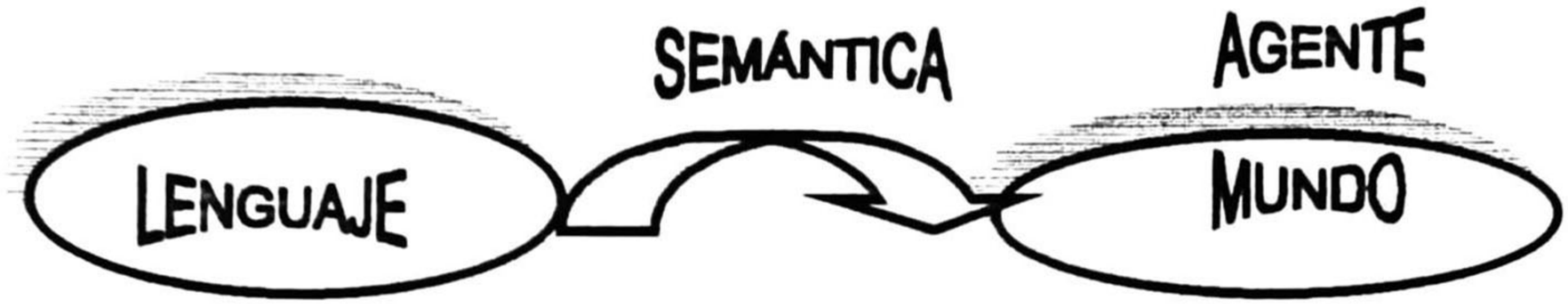


Fig. 3.1 Semántica del lenguaje

Pragmática: define la manera en que el lenguaje transforma los estados mentales de los comunicadores, indica como el lenguaje manipula la representación de los estados de un agente. Esto es representado en la fig. 5.2

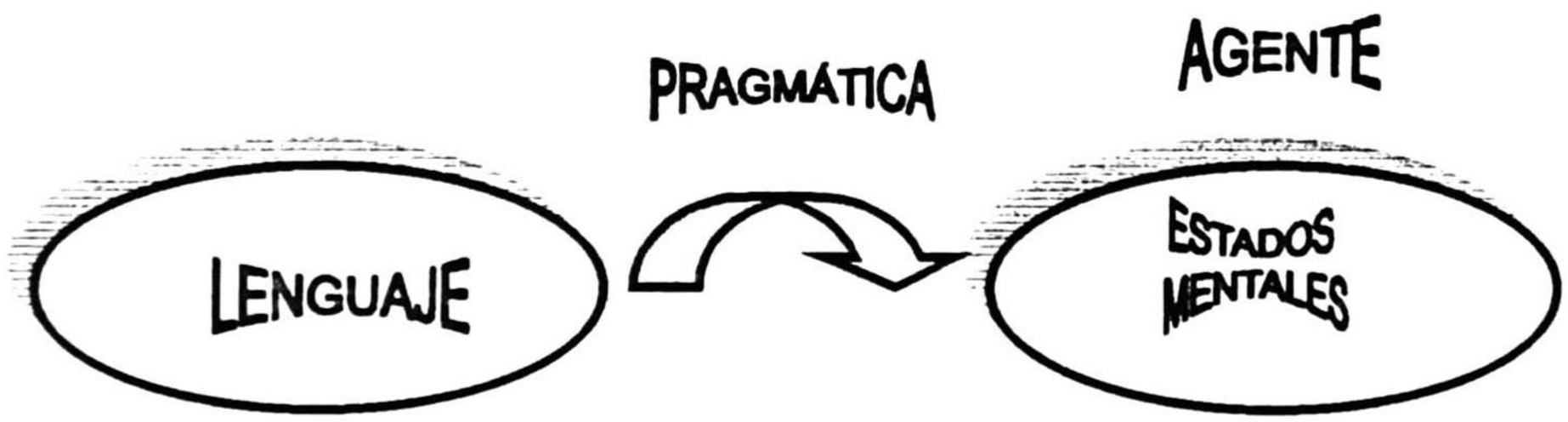


Fig. 3.2 Pragmática del lenguaje

Para lograr la interacción entre agentes se lleva a cabo, primero una comunicación para permitir un intercambio de mensajes entre los agentes, con los cuales es posible alcanzar un entendimiento y combinar esfuerzos para alcanzar los propósitos deseados (CSCW).

En la interacción se especifica el lenguaje que ha de utilizarse para establecer los diálogos con los cuales se lleve acabo una comunicación esperada. El lenguaje consta de un conjunto de acciones colectivas con las cuales los agentes realizan acciones o toman una decisión que ha sido influenciada por la presencia o conocimiento de otro agente. La fig. 3.3 representa el proceso

de la interacción y muestra la función de los lenguajes utilizados en el intercambio de mensajes entre agentes.
Para establecer la Interacción entre Sistemas MultiAgente se consideran los siguientes lenguajes.

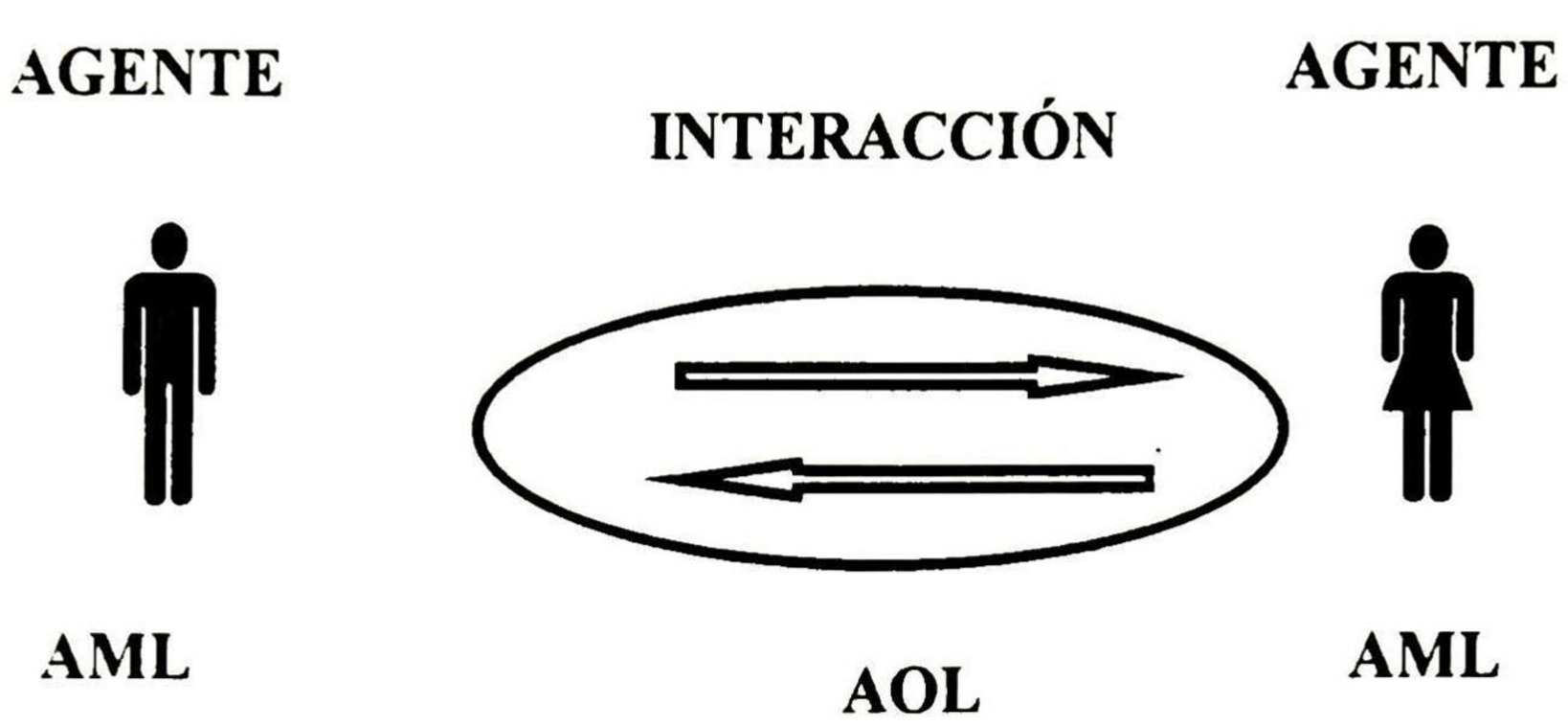


Fig. 3.3 Interacción entre agentes

AML es un metalenguaje formal con el cual se describe al agente por medio de un modelo matemático mediante el cual es posible especificar las propiedades y funcionalidad del agente. Con éste metalenguaje es factible modelar Sistemas MultiAgente y poder establecer su alcance, propiedades y su consistencia
AOL Es el lenguaje con el cual se logra la interacción entre los Sistemas MultiAgente, consta de un conjunto de sentencias expresadas en un lenguaje cotidiano, vistas como secuencias de mensajes o acciones compuestas de los mensajes de KQML[28]. Éste lenguaje corresponde al Lenguaje de "capacidad de Interacción" de Agentes en sistema Abiertos (LCIASA) que se presenta en el siguiente capítulo.

3.2 Lógica temporal de creencias

La Lógica Temporal de creencias es el modelo utilizado para describir a los agentes y corresponde al metalenguaje para describir al agente AML especificado en la fig. 3.3. Éste modelo considera el conocimiento que el agente tiene de sí mismo y de otros agentes (creencias) para llevar a cabo una acción o una secuencia de acciones.

3.2.1 Modelo de un Sistema MultiAgente:

El modelo de un MAS tiene las siguientes propiedades básicas: [31]

- Los agentes tienen nombres: Cada agente es identificado de manera única, por un identificador id del agente obtenidos del conjunto Ag.
- Los agentes tienen creencias: Se asume que las creencias del agente están expresadas en algún lenguaje interno L que puede ser un marco(trama) o un lenguaje de red semántica. Pero se supone que es un lenguaje lógico. Se escribe Form(L) para indicar al conjunto de fórmulas bien formadas (fbf) de L y sea BS el conjunto de creencias posibles que el agente puede tener, en otros términos:

$$BS \overset{def}{=} \wp(\text{Form}(L))$$

- Los agentes pueden ejecutar acciones: Se infiere que los agentes pueden ejecutar solamente acciones privadas, es decir pueden ejecutar acciones, las cuales operan en su propio estado. Sea Ac el conjunto de todas las acciones.
- Los agentes pueden enviar mensajes: Aunque las comunicaciones no se aplican de manera universal en DAI, esta es sin embargo una adjudicación común, entonces se supone que los agentes pueden comunicarse enviando mensajes. Un mensaje es una tripleta $< i, j, \phi >$ en donde $i \in$ Ag, representa al transmisor del mensaje, $j \in$ Ag es el receptor del mensaje y $\phi \in$ Form(L) es el contenido del mensaje. Sea Mess el conjunto de todos los mensajes:

$$\text{Mess} \overset{def}{=} \{ < i, j, \phi > \mid i, j \in \text{Ag} \wedge \phi \in \text{Form}(L) \}$$

Para especificar los mensajes recibidos por un agente, es de utilidad considerar una función rcv, de la siguiente manera:

$$\text{rcv}: \text{Ag} \times \wp(\text{Mess}) \rightarrow \wp(\text{Mess})$$

Tal que si $i \in$ Ag y m $\subseteq$ Mess entonces rcv(i, m) es el subconjunto de m en el cual i es el receptor.

Con esto es posible definir un agente como el cuádruplo:

$$A \overset{def}{=} < \beta^{o}, \mathcal{A}, M, \eta >$$

En donde:

$\beta^{o} \in BS$ es el conjunto de creencias iniciales del agente

$\mathcal{A} : BS \rightarrow Ac$ es la función de Acción del agente

$M : BS \rightarrow \wp(Mess)$ es la función de generación de mensajes del agente

$\eta : BS \times Ac \times \wp(Mess) \rightarrow BS$ es la función del siguiente estado del Agente.

Con esta definición es posible especificar las características del agente y observar su comportamiento a través de las acciones ejecutadas por el agente y de los cambios de estado que se presentan en el agente.
Tomando como base sus creencias iniciales, un agente selecciona una acción a ejecutar utilizando la función $\mathcal{A}$ y un mensaje a enviar, utilizando la función M La función η entonces cambia al agente de un estado a otro, tomando como base el mensaje recibido y la acción que este ha ejecutado.
Una vez especificado el concepto de agente, se define a un Sistema MultiAgente (MAS) como un conjunto indizado de tales agentes:

$$MAS \overset{def}{=} \{ < \beta^{o}_{i}, \mathcal{A}_{i}, M_{i}, \eta_{i} > : i \in Ag \}$$

3.2.2 El modelo de ejecución

El modelo de ejecución depende de la noción de estado de un sistema y de los cambios en el estado causados por las transiciones. Un estado es una instancia del conjunto de creencias de cada agente en el sistema en algún momento en el tiempo. Un cambio de estado o transición ocurre cuando un mensaje es recibido y se ejecuta una acción por parte de uno o más agentes.
El estado inicial del sistema está dado por el conjunto inicial de creencias del conjunto de agentes, esto es β^{o}_{i}

Formalmente, el estado β^{1}_{i} del agente i en el tiempo 1 está dado por la ecuación:

$$\beta^1_i \overset{\text{def}}{=} \eta_i(\beta^0_i, A_i(\beta^0_i), rcv(i, \cup_{j \in Ag} M_j(\beta^0_j)))$$

Este estado del agente i depende de la acción que i ejecutó, de los mensajes que le fueron enviados y de su estado inicial.
Esta ejecución puede generalizarse para proporcionar el conjunto de creencias del agente *i* para un tiempo arbitrario $U \in \mathbb{N}$ tal que $U > 0$

$$\beta^u_i \overset{\text{def}}{=} \eta_i(\beta^{u-1}_i, A_i(\beta^{u-1}_i), rcv(i, \cup_{j \in Ag} M_j(\beta^{u-1}_j)))$$

En el modelo de ejecución cada agente elige una acción a ejecutar, enviando mensajes, recibiendo mensajes, desplazándose hacia el siguiente estado y así consecutivamente.
De la ejecución del sistema se obtiene una historia de ejecución, la cual describe cada estado del agente, las acciones que este ejecuta y los mensajes que este envió en cada momento en el tiempo.
Sea Σ el conjunto de estas historias y σ un elemento de este conjunto, si denotamos por S_u al estado u-ésimo de σ y τ_u el evento que provoca la u-ésima transición de σ, podemos visualizar a σ como sigue:

$$\sigma : S_0 \xrightarrow{\tau_0} S_1 \xrightarrow{\tau_1} S_2 \rightarrow \ldots \xrightarrow{\tau_{u-1}} S_u \xrightarrow{\tau_u} \ldots$$

En ésta expresión se utiliza el símbolo S_u para denotar los estados de una historia, pero bien puede emplearse el Símbolo β, dada la analogía entre estados y creencias. Por otra parte se supone que la ejecución no termina.

3.2.3 Lógica temporal de creencias lineal en el tiempo (LA)

La lógica temporal es una rama de la lógica modal que trata con el desarrollo de situaciones en el tiempo, es decir describe situaciones dinámicas en donde la interpretación de las fórmulas se realiza usando marcos en los cuales las relaciones de accesibilidad son relaciones lineales.[30]

La lógica LA [31] es una lógica proposicional y contiene tres operadores atómicos: Bel para describir las creencias de un agente, Send para describir los mensajes que el agente envía y Do para describir las acciones que el agente ejecuta.

Adicionalmente LA contiene un conjunto de operadores modales temporales, los cuales permiten la descripción de las propiedades dinámicas de los agentes. Nótese que LA está basada en un modelo de tiempo que es lineal (cada momento en el tiempo tiene únicamente un sucesor), está acotada en el pasado(hay un inicio del tiempo) e infinita en el futuro (no hay un último momento en el tiempo).

3.2.3.1 Reglas sintácticas para LA

LA permite comprender respecto a los agentes: Sus creencias, sus acciones y los mensajes que envían. Se utilizan símbolos dentro del lenguaje que denotan agentes o acciones. Para expresar las creencias de los agentes, se define el lenguaje interno L de LA.

El lenguaje de LA basado en L contiene los siguientes símbolos:

- Los símbolos { True, Bel, Send, Do }
- Un conjunto contable de símbolos constantes tomados de los Conjuntos disjuntos $Cosnst_{Ag}$ (constantes de agentes) y $Const_{Ac}$ (constante de acción).
- Todas las fórmulas cerradas del lenguaje interno L
- El conector proposicional lógico unario $\neg$ y el conector binario $\vee$
- Los conectores temporales unarios { O, $\otimes$ } y los conectores temporales binarios {μ, ξ }
- Los símbolos de puntuación: {), (}.

La lógica LA está parametrizada por el lenguaje interno L. El conjunto de fórmulas (fbf) de LA basadas en el lenguaje interno L están definidas por las siguientes reglas:

1.- Si *i,j* son ids de agentes, ϕ es una fórmula cerrada de L y α es una constante de acción, entonces las siguientes son fórmulas de L:

true (Bel i, ϕ) Send(i, j, ϕ) (Do i, α)

2.- Si ϕ y ψ son fórmulas de LA, entonces las siguientes son fórmulas de LA:

$\neg\phi$ $\phi \vee \psi$

3.- Si ϕ y ψ son fórmulas de LA, entonces las siguientes son fórmulas de LA:

$O\phi$ $\otimes\phi$ $\phi \mu \psi$ $\phi \omega \psi$

La primera regla trata con fórmulas atómicas o átomos de AL, la fórmula true es una constante lógica para denotar verdad, esta siempre es satisfecha. La fórmula (Bel i, ϕ) se lee: el agente i cree ϕ. La fórmula (Do i, α) se interpreta como: el agente i ejecuta la acción α. La fórmula Send(i, j, ϕ) describe el envío de mensajes y será satisfecha si el agente i ha enviado al agente j un mensaje con el contenido ϕ.

Los conectivos proposicionales $\wedge$(and), $\vee$(or), $\Rightarrow$(if...then...), $\Leftrightarrow$(iff) y $\neg$(not) conserva su notación y semántica estándar. La fórmula $O\phi$ es satisfecha si ϕ es satisfecha en el siguiente tiempo (u+1). La fórmula $\otimes\phi$ es satisfecha si ϕ fue satisfecha en el tiempo anterior (u-1).

La tabla 1 presenta el significado de los operadores temporales con los cuales es posible escribir e interpretar correctamente expresiones para LA.

Regla	Significado
(Bel *i* ϕ)	El agente *i* cree ϕ
(Send *i j* ϕ)	El agente *i* envió al agente *j* el mensaje ϕ
(Do i α)	El agente i ejecuta la acción α
$O\phi$	El siguiente ϕ
$\otimes\phi$	Previamente ϕ
$\Box\phi$	De ahora en adelante ϕ
$\Diamond\phi$	Eventualmente ϕ

$\phi \mu \psi$	ϕ hasta que ψ (no estricto)
$\phi \omega \psi$	ϕ a menos que ψ
$\upsilon\phi$	ϕ hasta ahora
$\blacklozenge\phi$	ϕ alguna vez
$\alpha\phi$	ϕ antes
$\phi\beta\psi$	ϕ Cuando ψ
$\phi \Xi\psi$	ϕ desde que ψ

Tabla 1 Reglas esquemáticas para LA

3.2.3.2 Reglas semánticas para LA

La semántica proporciona el significado de los elementos de LA, la fig. 5.1 muestra las reglas semánticas para LA, indicando el significado de los operadores, con los cuales se dará una interpretación a las fórmulas especificadas en LA. [30]
Las fórmulas temporales se interpretan sobre un modelo M que es una sucesión infinita de estados M: $s_0, s_1, s_2, ...s_j...$
Un modelo para LA es una estructura del siguiente tipo: [32]

$$M = < \sigma, Ag, Ac, bel, action, sent, I >$$

Donde: $\sigma \in \Sigma$ es una historia de ejecución
Ag es el conjunto de ids para agentes
Ac es el conjunto de acciones
bel: Σ X Ag X $\mathbb{I} \rightarrow$ BS es una función que toma una historia de ejecución, un identificador de agente y un tiempo y retorna el conjunto de creencias del agente en la historia de ejecución en este tiempo.
action: Σ X Ag X $\mathbb{I} \rightarrow$ Ac es una función que toma una historia de ejecución, un identificador de agente y un tiempo y retorna una acción ejecutada por el agente en la historia de ejecución en ese tiempo.
sent: Σ X $\mathbb{I} \rightarrow \wp$(Mess) Es una función que toma una historia de ejecución y un tiempo y retorna el conjunto de mensajes enviados en la historia de ejecución en ese tiempo.
I: Constante $\leftrightarrow$ Ag $\cup$ Ac
Existe una estrecha relación entre el modelo formal del Sistema MultiAgente y los modelos lógicos para LA, se puede caracterizar esta relación

estableciendo las condiciones bajo las cuales un modelo para LA puede ser considerado para representar una ejecución del modelo multiagente.

Un modelo

$$M = < \sigma, Ag, Ac, bel, action, sent, I >$$

para una LA representa una corrida del Sistema MultiAgente:

$$MAS \overset{def}{=} \{ < \beta^{o}_{i}, A_{i}, M_{i}, \eta_{i} > : i \in Ag \}$$

Es decir, tomando el MAS a partir de su estado inicial y considerando sus cambios de estado, obtenemos el modelo M, si se cumplen las siguientes condiciones:

$$\forall u \in \; \forall i \in Ag \; bel(\sigma, i, u) = \beta^{u}_{i} \; \wedge action(\sigma, i, u) = A_{i}(\beta^{u}_{i}) \; \wedge \; sent(\sigma, u) = \cup_{j \in Ag} Mj(\beta^{u}_{j})$$

La relación de satisfacibilidad (|=) para la LA se establece entre pares de la forma < M, u> y fórmulas de LA. Dado un modelo M y una fórmula temporal ϕ se presenta la noción de que ϕ se cumple en una posición $u \geq 0$ de M y lo que denotamos por : $< M, u> \models \phi$

Los conectivos utilizados son :

$\neg$ (not) $\wedge$ (and) $\vee$ (or) $\Leftrightarrow$ (iff) $\Rightarrow$ (if...then)

Reglas semánticas para LA:

Para dar representación y una interpretación a las expresiones del lenguaje LA y para verificar que la especificación de un Sistema MultiAgente es correcta, se utilizan las reglas semánticas que se presentan en la fig. 3.1. [30]

$< M, u> \models$ true

$<M, u> \models (Bel\ i, \phi)$ iff $\phi \in bel(\sigma, I(i), u)$

$< M, u> \models (Do\ i, \alpha)$ iff $action(\sigma, I(i), u) = I(\alpha)$

$< M, u> \models (Send\ i, j, \phi)$ iff $< I(i), I(j), \phi> \in sent(\sigma, u)$

$< M, u> \models \neg\phi$ iff $< M, u> \not\models \phi$

$< M, u> \models \phi \vee \psi$ iff $< M, u> \models \phi$ or $< M, u> \models \psi$

$< M, u> \models O\phi$ iff $< M, u+1> \models \phi$

$$< M, u> \models \otimes\phi \text{ iff } u>0 \text{ and } < M, u-1> \models \phi$$
$$< M, u> \models \Box\phi \text{ iff } \forall k \geq u \quad < M, k> \models \phi$$
$$< M, u> \models \Diamond\phi \text{ iff } \exists k \geq u \quad s.t < M, k> \models \phi$$
$$< M, u> \models \phi \mu\psi \text{ iff } \exists k \in N \text{ s.t. } k \geq u \text{ y} < M, k> \models \psi \text{ and}$$
$$\forall w \in N \text{ s.t } u \leq w < k, < M, w> \models \phi$$
$$< M, u> \models \phi\omega\psi \text{ iff} < M, u> \models \phi \mu\psi \text{ ó } < M, u> \models \Box\phi$$
$$< M, u> \models \upsilon\phi \text{ iff } \forall 0 \leq k \leq u \quad < M, k> \models \phi$$
$$< M, u> \models \blacklozenge\phi \text{ iff } \exists 0 \leq k \leq u \quad < M, k> \models \phi$$
$$< M, u> \models \alpha\phi \text{ iff } < M, 0> \models \phi \text{ ó } < M, u> \models \otimes\phi$$
$$< M, u> \models \phi\beta\psi \text{ iff } < M, u> \models \phi\xi\psi \text{ ó } < M, u> \models \upsilon\phi$$
$$< M, u> \models \phi \xi\psi \text{ iff } \exists 0 \leq k \leq u \text{ s.t} < M, k> \models \psi \text{ and}$$
$$\forall w \in N \text{ s.t } k < w \leq u \quad < M, w> \models \phi$$

Fig.3.1 Reglas semánticas para LA

A continuación se da una interpretación gráfica de algunas de las reglas semánticas de la Fig. 3.1:

La regla esquemática para el operador "hasta que " ($\phi\mu\psi$), la cual esta dada por la siguiente expresión semántica:

$$< M, u> \models \phi \mu\psi \text{ iff } \exists k \in N \text{ s.t. } k \geq u \text{ y} < M, k> \models \psi \text{ and}$$
$$\forall w \in N \text{ s.t } u \leq w < k, < M, w> \models \phi$$

Se representa en forma gráfica, en la fig. 5.2:

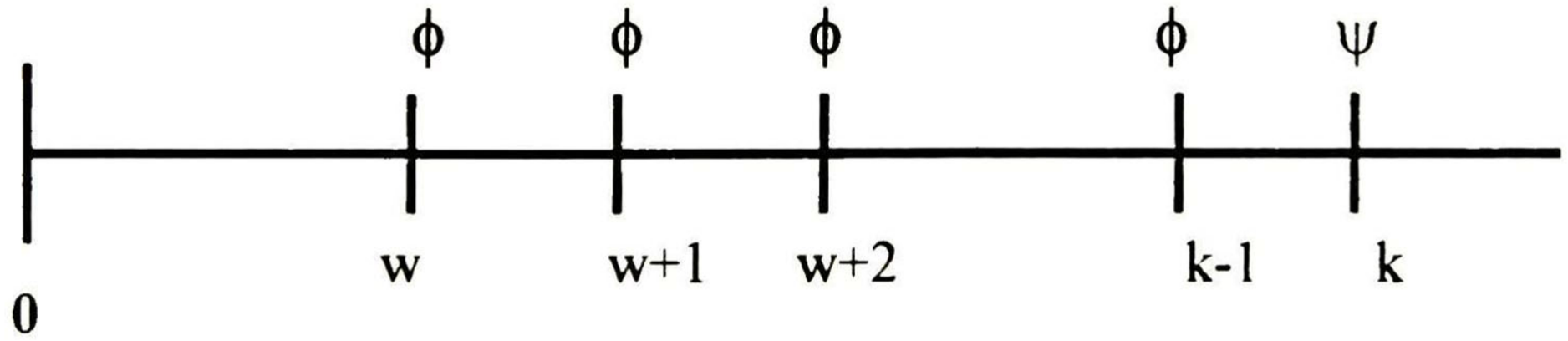


Fig. 3.2 Representación gráfica de "hasta que"

En su interpretación dice que $\phi \mu\psi$ es satisfecha si ϕ es satisfecha en todo momento hasta que ψ se satisfaga. En esta regla ψ debe ser satisfecha eventualmente.

La regla esquemática " alguna vez" ($\blacklozenge\phi$) dada por la expresión:

$$< M, u> \models \blacklozenge\phi \text{ iff } \exists\ 0 \leq k \leq u \ < M, k> \models \phi$$

Tiene una representación gráfica dada por la Fig. 3.3:

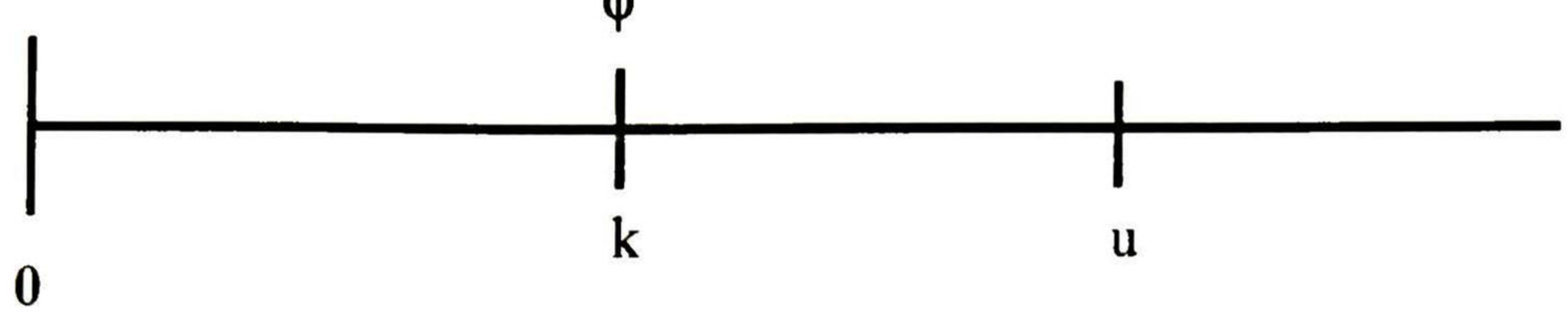


Fig. 3.3 Representación gráfica de "alguna vez"

En su interpretación dice que $\blacklozenge\phi$ es satisfecha si ϕ se cumplió en algún instante anterior o igual a u.
La regla esquemática "desde que" ($\phi\ \xi\psi$) dada por la expresión semántica:

$$< M, u> \models \phi\ \xi\psi \text{ iff } \exists\ 0 \leq k \leq u \text{ s.t} < M, k> \models \psi \text{ and}$$
$$\forall\ w \in \_\tilde{} \text{ s.t } k < w \leq u \ < M, w> \models \phi$$

Tiene la representación gráfica que ilustra la fig. 5.4

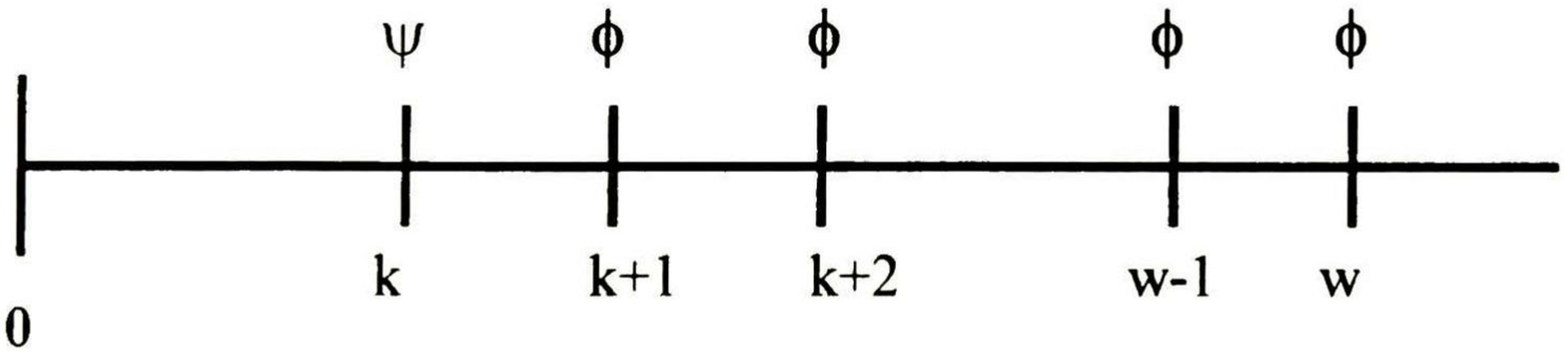


Fig. 3.4 Representación gráfica de "desde que"

En su interpretación dice que $\phi\xi\psi$ es satisfecha si ϕ se cumple en todo momento desde que ψ sea satisfecha. En esta regla ψ debe ser satisfecha eventualmente. En otras palabras, primero debe cumplirse ϕ y después ψ.
Una vez desarrollada la lógica, la cual puede ser usada para representar las propiedades de los Sistemas MultiAgente, se consideran las formas para su utilización, hay dos formas en que puede usarse la lógica: para *especificación* y para *verificación*.

3.3 Especificación y Verificación

3.3.1 Especificación

La especificación de un sistema es una que descripción de las propiedades que se pretende que el sistema debe exhibir. Una especificación formal es aquella que es expresada (en su mayor parte) en un lenguaje matemático, como consecuencia se puede usar LA como un lenguaje de especificación formal para Sistemas MultiAgente. Esto es, nuestro lenguaje tiene contemplada esta etapa, de aquí su innovación y potencial.
Existen dos propiedades que se desea especificar para los sistemas que responden a reacciones:

- *Vivacidad*: Propiedad que asevera que algo bueno va a pasar.
- *Seguridad*: Propiedad que afirma que nada malo pasa

Se aplica la lógica LA para especificar de manera formal el protocolo de comunicación basado en el speech-Act, para alcanzar la parte del entendimiento en el proceso de comunicación entre agentes, bajo un esquema de actividad cooperativa.

Tomar como ejemplo un protocolo simple para la actividad cooperativa, en este protocolo los agentes se comunican por medio de mensajes, en este caso tres:

- Request(α) Una petición para que la acción α sea llevada
- Abort(α) El Transmisor informa al receptor que rechaza la acción α
- Inform (ϕ) El transmisor informa al receptor del hecho ϕ

Se asumen dos predicados de dominios

Friend (i) El agente i es amigo
Trust (i) El agente i es acreditado

Un agente que recibe un Request de un agente amigo para hacer alguna acción, la hará eventualmente, pero no lo hará de otro modo (las acciones solo serán hechas por la petición de un agente amigo). Si un agente recibe un mensaje Inform, este agregará el hecho a las creencias, solo si el emisor está acreditado.

Este protocolo simple puede especificarse con facilidad, primero consideremos que un agente que recibe un Request es un agente amigo, entonces hará una acción, por la propiedad de vivacidad. Se especifica de la siguiente manera:

$$\Box\forall i.\forall j.\forall\alpha.(\text{Send } i, j \text{ Request}(\alpha)) \wedge(\text{Bel } j \text{ Friend(i)}) \Rightarrow \Diamond(\text{ Do } j\,\alpha)$$

Siempre se cumple que si i envia a j un Request para α y j cree que i es amigo, entonces eventualmente se ejecuta α

Lo siguiente expresa la propiedad de seguridad, es decir que si un agente hace alguna acción entonces la ejecución de la acción debe estar precedida por un Request por parte del agente amigo, para ejecutar la acción.

$$\Box\forall i.\forall j.\forall\alpha.[\ \neg\ ((\text{Send } i, j \text{ Request}(\alpha)) \wedge(\text{Bel } j \text{ Friend(i)}))$$
$$\xi$$
$$(\text{Do j } \alpha\ ) \wedge \neg\ (\text{Send } i, j \text{ Request}(\alpha))]\ \Rightarrow \neg\ (\text{Do j } \alpha\ )$$

La siguiente propiedad de vivacidad afirma que un agente que recibe un mensaje Inform por parte de un agente acreditado, llegará eventualmente a creer el contenido del mensaje.

$$\Box\forall i.\forall j.\ (\text{ Send } i\,j \text{ Inform } (\phi)) \wedge (\text{ Bel } j \text{ trusted}(i))$$
$$\Rightarrow\Diamond(\text{Bel } j\,\phi)$$

El modelo se generaliza tomando las acciones α como parte del conjunto Ac (constantes de Acción), para todas las acciones expresadas en los mensajes considerados en el Speech-Act: [28].

< speech-act> ::= (< inf-act-stat>|< BD-act-stat>|< Asn-act-stat>
|<Req-act-stat>|<Mresp-act-stat>|<Def.cap-act-stat>
|<Notif-act-stat>|<Conect-act-stat>|<Fac-act-stat>)

3.3.2 Verificación

Así como es posible utilizar LA para la especificación es también posible usarla para la verificación. Generalmente la verificación es considerada como un proceso mucho más complejo que la especificación.
La verificación es el proceso de mostrar que un sistema implementado es correcto con respecto a su especificación. Formalmente la verificación de un sistema involucra la prueba en el sentido matemático, de que es correcto.
Una de las pruebas de verificación consiste aplicar arboles semánticos modales para mostrar la satisfacibilidad de los modelos que representan a los procesos del sistema.

Conclusión: En este capítulo se presenta el Metalenguaje de Agentes (AML) con el cual es posible describir las propiedades que un Sistema MultiAgente debe exhibir. La importancia de este lenguaje es su potencial para estudiar las propiedades, funcionalidad y consistencia de los Sistemas MultiAgente, antes de pasar a la fase de implementación .
El Metalenguaje AML esta fusionado al lenguaje LCIASA que se propone, en el sentido de considerar al Metalenguaje AML como una etapa previa (Análisis y diseño) a la instrumentación de un Sistema MultiAgente. Una vez diseñado y modelado el SMA es utilizado el lenguaje LCIASA para implementar la aplicación. Esta asociación hace posible utilizar el lenguaje LA como herramienta formal para diseñar y modelar Sistemas MultiAgente.

Capítulo 4

LCIASA: Lenguaje de "Capacidad de Interacción" de Agentes en Sistemas Abiertos

Objetivo: Este capítulo tiene por objetivo presentar el lenguaje con el cual los agentes puedan establecer un diálogo o interacción (LCIASA), Para lo cual se exhibe el paradigma que genera al lenguaje, su sintaxis, semántica, la arquitectura del agente y los protocolos de interacción asociados .

El lenguaje LCIASA[1] es un lenguaje para el desarrollo de aplicaciones multiAgente es decir aplicaciones cooperativas, el lenguaje está conformado por un conjunto de sentencias, que ayudan a realizar un plan, mediante un intercambio de mensajes. Mediante el intercambio de mensajes los agentes inmersos en un sistema abierto (sistema distribuido en donde no hay restricción de acceso a la información), negocian para lograr un posible entendimiento entre ellos y alcanzar la conclusión del plan o tarea específica .

LCIASA es un lenguaje de comunicación entre agentes (ACL), esta clasificado como un lenguaje de comunicación basado en el modelo de comunicación humana.

En nuestro caso el concepto "Capacidad de Interacción" se refiere a la habilidad que tiene los agentes para realizar una tarea determinada (un plan).

La "capacidad de Interacción" se refleja en la secuencia de acciones con la cual el agente puede ejecutar una Intención o tarea determinada. Con ésta secuencia de acciones es posible cuantificar la expresividad de las sentencias del lenguaje, es decir se puede estimar la complejidad y alcance de la tarea realizada.

Las sentencias de LCIASA están expresadas en un lenguaje cotidiano, "mas humano", comprensible, evidente para una persona con escasos conocimientos de la computación; con el fin de facilitar la comprensión e instrumentación de aplicaciones de sistemas multiAgente.

---

[1] LCIASA Lenguaje de "Capacidad de Interacción " de Agentes en Sistemas Abiertos

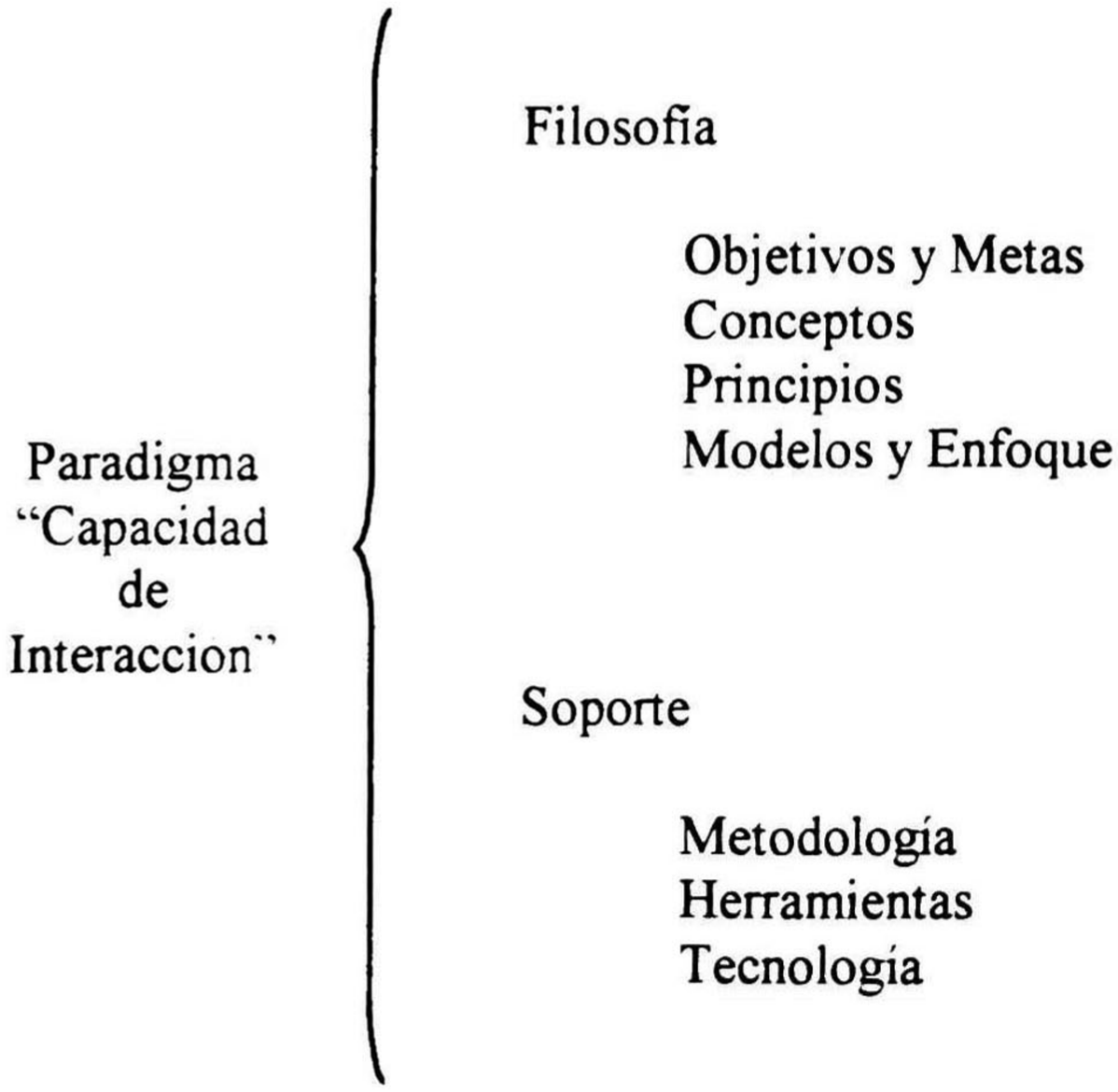
4.1 El paradigma "Capacidad de Interacción"

El concepto de paradigma es considerado como un modelo de pensamiento en el que se da una visión distinta de la realidad. En el contexto de la computación un paradigma y los lenguajes de programación que lo soportan forman una asociación. Un paradigma engendra los lenguajes y estos a su vez existen como parte integral del paradigma.

El paradigma "Capacidad de Interacción" considera a un conjunto de sentencias utilizadas por un agente para cooperar con otros agentes, aquí se le asigna un significado particular a la palabra "Capacidad de Interacción", con la cual se expresa la habilidad que tienen los agentes para realizar una secuencia de operaciones que denominamos planes, con los cuales llevar a cabo una tarea específica.

Los Planes dados por las secuencias de acciones asumen tanto un *conocimiento perfecto* (serie de acciones sin incertidumbre)por parte de los agentes, como planes con información parcial (sucesión de acciones con *incertidumbre).*

El paradigma "Capacidad de Interacción " es concebido como un modelo basado en el intercambio de planes, en el que se da una visión particular de la forma de interactuar entre agentes de software inmersos en un ambiente distribuido abierto (los recursos son compartidos por todos y cada uno de los dispositivos que componen el sistema). El paradigma esta definido por los siguientes conceptos:



Los Sistemas MultiAgente ofrecen una mayor ventaja por la aptitud que muestran para elaborar planes, aún cuando se tenga incertidumbre en el curso de acciones a seguir. Es decir el agente obtiene cierta información, la cual le sirve de base para elaborar un plan que le permita ejecutar una tarea determinada. Si se requiere reducir la incertidumbre, el agente trata de conseguir una mayor información para realizar la tarea con una mayor precisión.

El paradigma tiene por una parte un enfoque evolutivo ya que toma como base el conjunto de mensajes de KQML, con los cuales se forman secuencias de acciones para formar las sentencias del lenguaje de interacción entre agentes.

El conjunto de sentencias del lenguaje de interacción entre agentes tiene como finalidad lograr la interacción entre los agentes inmersos en un ambiente abierto, en donde se tenga transparencia tanto en las plataformas de trabajo como en las aplicaciones que se ejecuten. El lenguaje contiene sentencias que garantizan la seguridad e integridad de la información al trabajar en este tipo de ambientes.

4.1.1 La filosofía del paradigma

La filosofía es considerada como la doctrina que impone como único criterio de valoración de todo principio teórico sus efectos, consecuencias o resultados prácticos, es decir que el contenido inteligible y la evaluación de una experiencia únicamente consisten en los efectos que se obtienen de ella. En nuestro caso consideramos la filosofía como el Sistema de principios que se establecen para explicar o agrupar ciertos hechos.

Objetivos
- Legibilidad en el código
- Transparente a las aplicaciones
- Independiente de las plataformas de operación

El lenguaje LCIASA esta basado en sentencias del lenguaje cotidiano con lo cual se persigue alcanzar una *legibilidad* al momento de codificar las aplicaciones, las cuales son especificadas por mensajes cuyo contenido esta escrito en diversos formatos, con lo cual obtenemos aplicaciones que son

independientes de las *tareas* que se están ejecutando y de las *plataformas* en que los dispositivos estén operando.
La Legibilidad en el código hace alusión a la facilidad de interpretación del código utilizado para desarrollar una aplicación, como consecuencia del lenguaje diáfano utilizado para expresar sus sentencias.
La transparencia a las aplicaciones indica la autonomía que el lenguaje tiene respecto al tipo de aplicaciones que se estén ejecutando en un momento dado dentro o fuera del sistema en donde se ejecuta la aplicación.

Metas
- Facilitar la Instrumentación de sistemas complejos
- Desarrollar aplicaciones confiables en sistemas abiertos
- Facilitar la instrumentación de aplicaciones de Sistemas MultiAgente
- Alcanzar una adaptación en las aplicaciones actuales

La instrumentación de aplicaciones distribuidas es una labor complicada con el conjunto de acciones y de mensajes que forman parte de LCIASA, el encargado de elaborar aplicaciones cuenta con una herramienta que le facilita *la instrumentación de aplicaciones complejas.*
Dado que el lenguaje cuenta con sentencias para confirmar que los mensajes lleguen al destino correcto y que las acciones se ejecuten de manera correcta, se busca alcanzar el desarrollo de *aplicaciones confiables en sistemas abiertos.*
Como consecuencia de lo mencionado anteriormente, LCIASA *facilita la instrumentación* de aplicaciones MultiAgente y hace posible alcanzar una *adaptación a las aplicaciones actuales.*

Conceptos
- Mensaje
- Plan
- Protocolo de interacción asociado

El mecanismo de interacción entre los agentes se realiza por medio del intercambio de *mensajes,* los cuales son agrupados para formar un *plan,* éste es conceptuado como una secuencia de mensajes para instrumentar una tarea determinada, a esta secuencia se le denomina de manera análoga *protocolo de interacción asociado.*

Los protocolos de interacción asociados son utilizados para mostrar en forma gráfica, a los agentes que intervienen en la consecución de una tarea específica, así como la secuencia de acciones para llevar a cabo ésta labor.
El entendimiento entre los agentes es alcanzado por medio de un intercambio de mensajes (*pase de mensajes*) con los cuales es posible elegir al agente destino, autentificar en mensaje y asegurar que las acciones correspondientes a los mensajes se realicen de manera correcta.

Principios
- Pase de mensajes
- Mensajes "Head-Body"
- Independencia de contexto
- Capacidad de Interacción

Los mensajes estan encapsulados conteniendo una parte en la cual se especifican los parámetros asociados al proceso de comunicación llamado "*Head*" y otra parte en la cual se encuentra comprendido el mensaje llamada "*Body*", en donde el contenido del mensaje es expresado en diversas ontologias, conduciendo a tener aplicaciones *independientes del contexto.*
Como se mensionó al principio del capítulo la expresividad en las sentencias, es decir la habilidad que tienen los agentes para llevar a cabo planes, conduce al principio de "*Capacidad de Interacción*" en el cual se basa este paradigma.

Modelos
- Estímulo – Reacción
- Colaboración
- Máquinas Virtuales

Los agentes *responden* a *estímulos* dados por los mensajes que son intercambiados, por medio de la permuta de estos mensajes se logra la *colaboración* necesaria para ejecutar operaciones, las cuales son asignadas por medio de interfaces con el usuario y con el dispositivo en donde se ejecutan las tareas asignadas, a estas interfaces se les conoce como *máquinas virtuales*.

Enfoque evolutivo. El paradigma esta sujeto a los estándares proporcionados por DARPA[28]. Éste está Basado en la funcionalidad de KQML, por lo cual LCIASA es considerado como un *dialecto* o extensión de KQML.

Enfoque { Evolutivo / Revolucionario

Enfoque revolucionario. El paradigma agrupa mensajes para realizar tareas específicas, de acuerdo a un plan previamente establecido para ejecutar las tareas deseadas. Con el paradigma "Capacidad de Interacción" Es posible realizar aplicaciones en sistemas distribuidos abiertos robustos ya que dentro de las primitivas se proporcionan sentencias que aseguren la fiabilidad de las aplicaciones.
La parte medular del enfoque revolucionario se da por el hecho de contar con el lenguaje AML asociado al lenguaje LCIASA para obtener un modelo formal de la especificación del Sistema MultiAgente. Éste modelo puede ser utilizado como una herramienta de diseño, para conocer el comportamiento del sistema, antes de implementarlo como un sistema informático.

4.1.2 El Soporte

El soporte toma las herramientas de apoyo que proporcionan a un paradigma los medios tangibles para llegar a su realización, éstas herramientas permiten concretar la filosofía asociada a un paradigma, dando como resultado el desarrollo completo e integral del paradigma en discusión.
La metodología abarca: la definición de los conceptos de agentes y multiAgentes, del modelo formal que describe los Sistemas MultiAgente , el paradigma que define al lenguaje, la arquitectura del agente y el lenguaje para definir la interacción en Sistemas MultiAgente.
El paradigma toma a KQML, Java, C++, CORBA como las *herramientas* principales para Implantar el lenguaje. Como se menciono anteriormente KQML ofrece un conjunto de acciones primitivas que son la base de las sentencias de LCIASA. Java y C++ se utilizan como lenguajes para el desarrollo de las interfaces del ORB (mensajero de solicitudes de objetos) de

CORBA que es la plataforma de desarrollo que tomamos como base para dar la transparencia que requerimos en nuestras aplicaciones.
Asimismo el paradigma toma la *tecnología* de la Internet con sus protocolos asociados como el cimiento para transportar la información.

4.2 Arquitectura del agente

En la arquitectura del agente para el lenguaje de interacción entre agentes basado en la "capacidad de Interacción" (LCIASA) se definen los mecanismos que permitirán pasar del modelo teórico a la práctica, además se consideran los componentes que posibilitan la construcción de Sistemas multiAgente, de tal forma que satisfagan las especificaciones dadas por los modelos teóricos.
La arquitectura de LCIASA puede considerarse como un subconjunto de la arquitectura BDI, que a su vez son incluidas, como un caso especial en la arquitectura ICE (abreviación de $I^2C^2E^2$: Información, Intención, Comunicación, Cooperación, Evaluación, Empowerment) y esta es usada para especificar la interacción entre agentes dinámicos. Como se ilustra el la fig. 4.1

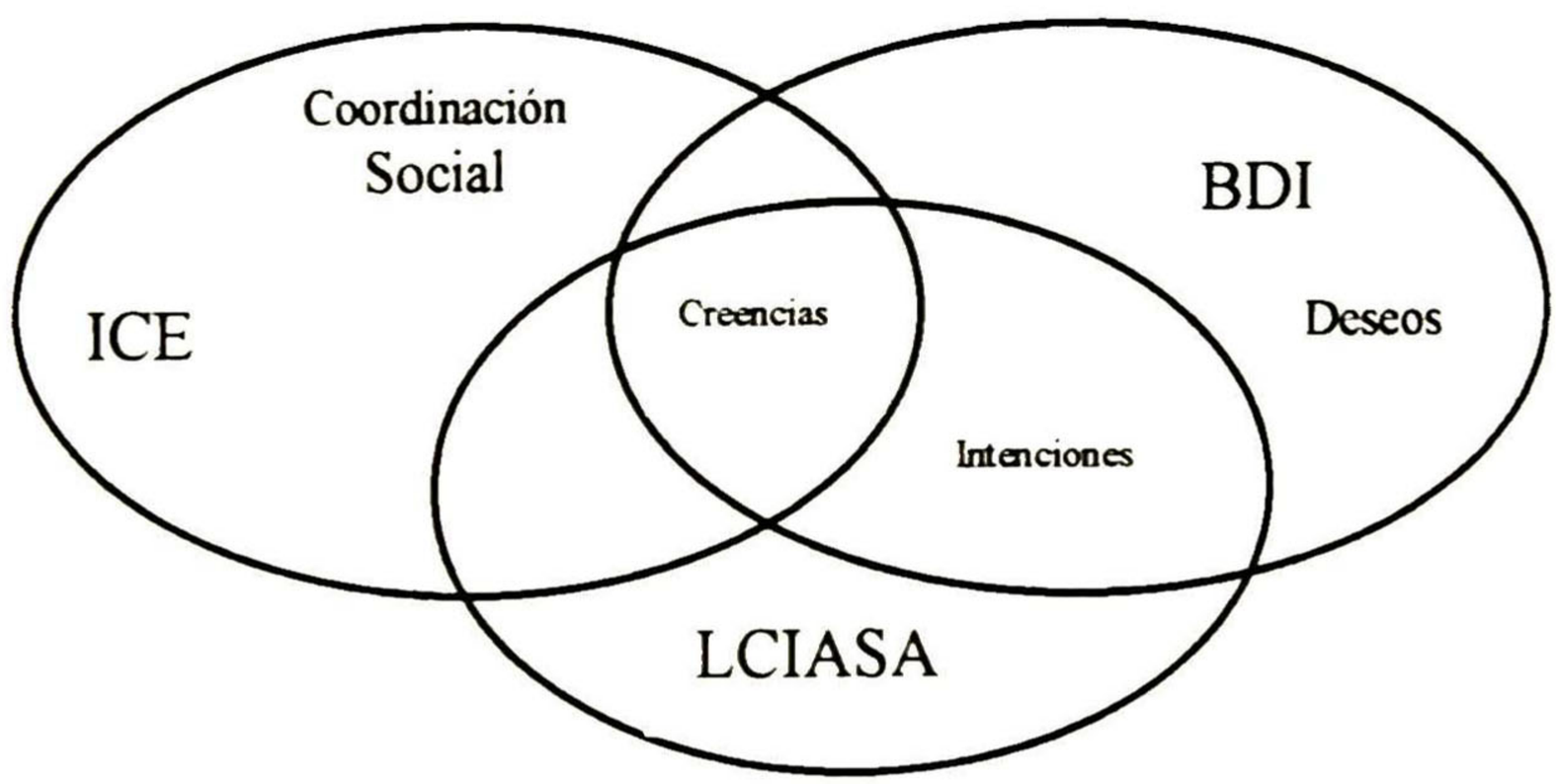


Fig. 4.1: contexto de la arquitectura para el lenguaje LCIASA

El tipo de arquitectura considerado para el LCIASA corresponde al tipo de arquitectura clásica llamada arquitectura deliberativa, que es aquella que contiene un modelo simbólico explícito del mundo, en el cual las decisiones

son hechas por medio de un razonamiento lógico basado en el empate de patrones y en la manipulación simbólica.
Esta arquitectura está basada en creencias, e intenciones, contiene las siguientes estructuras de datos clave:
Una *librería del plan.* Ésta contiene una representación explícita de creencias, e intenciones.
Adicionalmente la arquitectura contiene un *razonador,* que le permite razonar acera del mundo, un analizador de significado terminal, para determinar cuales planes pueden ser usados para alcanzar las intenciones del agente.

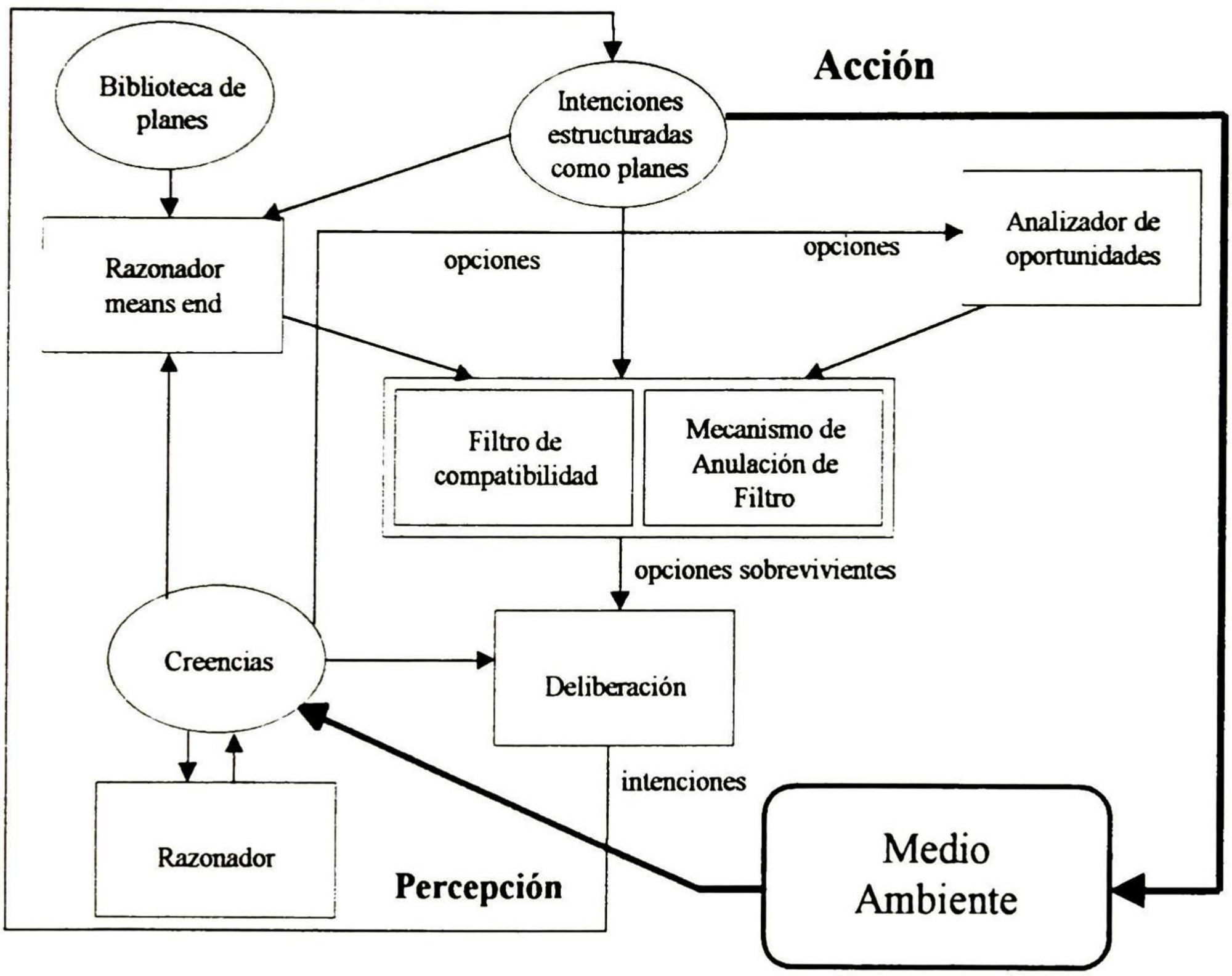


Fig. 4.2: Arquitectura del Agente

El razonador "means-end" se invoca para cada uno de los planes parciales, para proponer subplanes que lo completen, un *analizador de oportunidades*

que percibe el medio ambiente con el fin de determinar opciones posteriores para el agente, un *proceso filtrador* y un *proceso deliberador,* el proceso filtrador es responsable de determinar el subconjunto de los cursos de acciones potenciales del agente, que tienen la propiedad de ser consistentes con las intenciones actuales del agente, la elección entre las opciones que compiten es hecha por el *proceso de deliberación* el cual recibe las opciones sobrevivientes después del filtrado y pondera las opciones unas contra otras y actualiza las intenciones estructuradas como planes.

El *Almacén de creencias* contiene una representación de lo que el agente conoce de sí mismo y de su medio ambiente, como se muestra en la Fig. 4.2

Considerando la arquitectura mostrada en la fig. 4.2, y con la finalidad de dar una mayor claridad en cuanto a los componentes y funciones de los mismos, se presenta la fig. 4.3 en donde se esquematiza la estructura de un agente en la ISM (Interacción en Sistemas Multiagente), la cual puede conceptuarse como una arquitectura modular, de la siguiente forma:

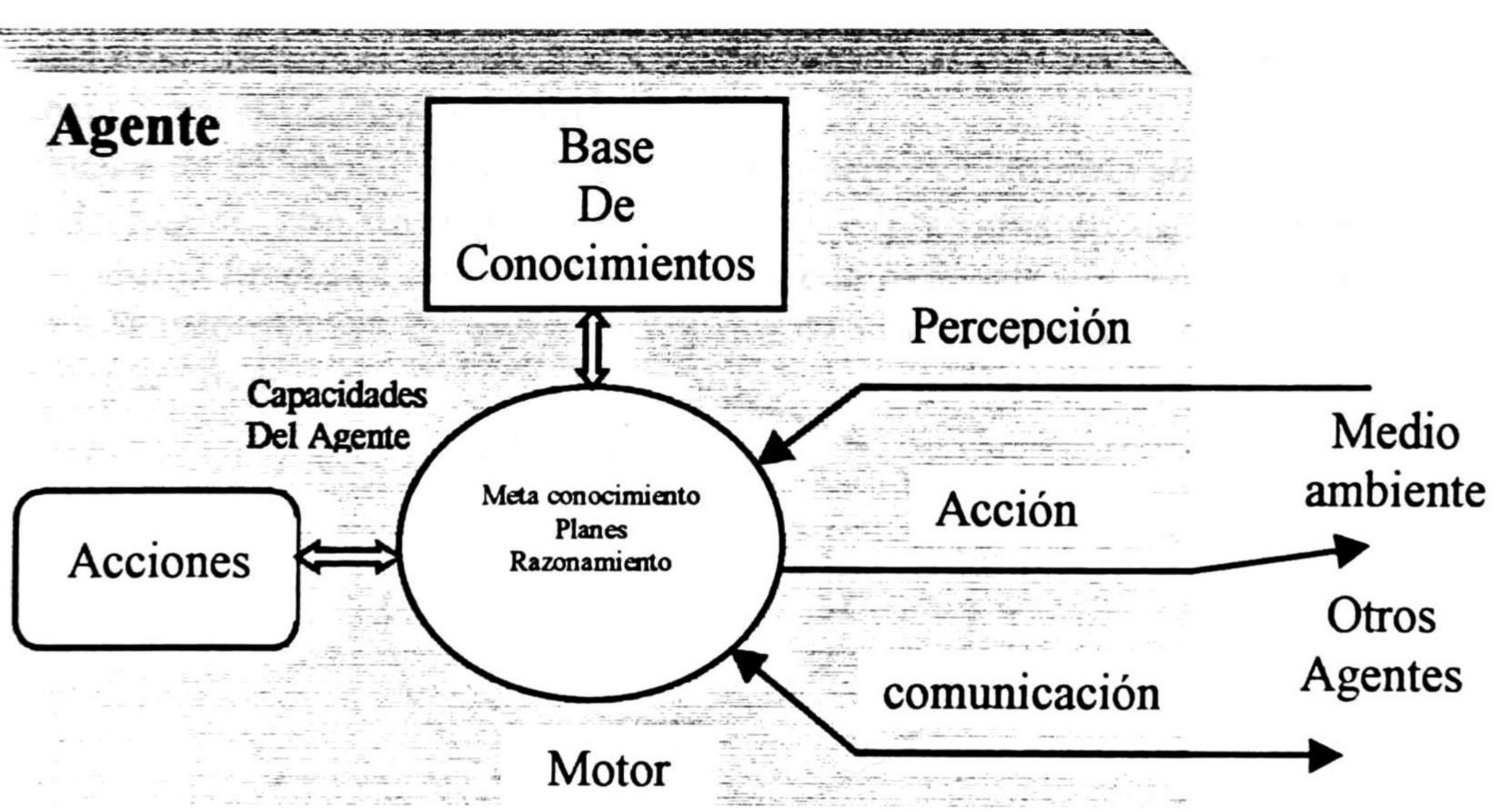


Fig. 4.3 Estructura de un agente

En donde:

La base de conocimientos mantiene el conocimiento del agente mismo, de su medio ambiente externo y de otros agentes.
Las acciones representan los medios del agente para percibir el mundo externo, actuar sobre el mundo externo y comunicarse con otros agentes. Las acciones pueden dividirse en acciones primitivas y complejas (acciones definidas por la composición de otras acciones).
El motor define el comportamiento del agente, esto es la forma de actuar en respuesta a su conocimiento y a las interacciones con el mundo externo.

4.3 Lenguaje de "Capacidad de interacción" de Agentes en Sistemas Abiertos (LCIASA)

LCIASA corresponde al lenguaje Objeto de agentes (AOL), como se mostró en la fig. 2.3. El lenguaje objeto describe la interacción en los Sistemas MultiAgente. El lenguaje LCIASA consta de un conjunto de sentencias basado en las acciones atómicas de KQML (speech-act). La fig. 4.4 muestra la dependencia entre el LCIASA y KQML
Las sentencias de LCIASA se consideran como acciones compuestas de los mensajes de KQML, con un objetivo predefinido, expresadas en un lenguaje coloquial y con la finalidad de que el usuario pueda expresar sus necesidades de acceso a la información de una manera simple y adecuada a la forma de expresarse cotidianamente.

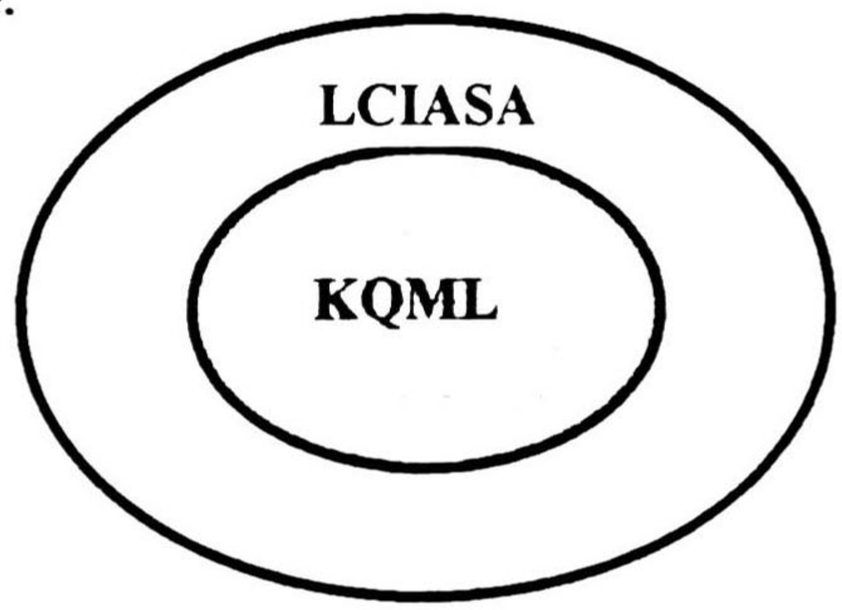


Fig. 4.4 Subordinación de LCIASA respecto a KQML

Con el conjunto de mensajes se tiene acceso a la información almacenada en un grupo de dispositivos interconectados bajo un esquema distribuido abierto, en donde se da una interacción entre agentes para acceder a la información

independientemente del contexto, tecnología o plataforma empleada en cada uno de los componentes del sistema.

4.3.1 Sintaxis de LCIASA

La sintaxis es considerada como el conjunto de reglas que generan las sentencias del lenguaje, para su especificación se utiliza la notación BNF.
El LCIASA está basada en un lenguaje de comunicación entre agentes, orientado a sentencias, las cuales constan de un conjunto de mensajes reconocidos por un identificador y por un conjunto de parámetros, con los cuales se señala la función que realizan. En nuestro caso los mensajes corresponden a las acciones primitivas de KQML.

La sintaxis de un mensaje expresada en BNF es la siguiente:

<mensaje>::=(<palabra>{<blanco>:<palabra><blanco><expresión>}*)

<expresión>::= <palabra> | <quotation> | <cadena> |
(<palabra> {<blanco> <expresión>}*)

<palabra>::= <caracter> <caracter>*

<caracter>::= <alfabético> | <numérico> | <especial>

<especial>::= < | > | = | + | - | * | / | & | ^ | ~ | _ | @ | $ | % | : | . | !
| ?

<quotation>::= ´<expr> | `<comma-expr>

<comma-expr>::= <palabra> | <quotation> | <cadena> |
,<comma-expr> | (<palabra> {<blanco> <comma-expr>}*)

<cadena>::= "<cadcaracteres>*" | #<digito><digito>*
"<ascii>*

<cadenacaracteres>::= \<ascii> | <ascii>-\-<"">

En la capa de comunicaciones los agentes intercambian "mensajes" (packages), los cuales contienen un mensaje empaquetado en donde se especifican algunos atributos de comunicación. Un "mensaje" es representado como una lista cuyo primer elemento el átomo "package" y el resto de sus elementos son atributos

que utilizan el formato de KQML para expresar sus valores. El paquete tiene una estructura "Head - Body" como se ilustra a continuación:

```
(PACKAGE : FROM aplic10
           : TO aplic11
           : ID  IDV-g001-111234.10115131
           : COMM sync
```

} Head

```
: CONTENT
          (MSG
               : TYPE query
               : Qualifiers ( :NUMBER-ASNWER 1)
               : CONTENT-LANGUAJE KIF
               : CONTENT (color snow-C)))
```

} Body

En el ejemplo la aplicación 10 le envía una consulta a la aplicación 11.
Tomando en cuenta la función que los "mensajes" llevan a cabo, estos se agrupan en los siguientes tipos:

Informativos	Notificación	Consulta
Base de Datos	Conectividad	Multirespuesta
Respuesta	Facilidades	De efecto

De esta forma las acciones de comunicación pueden expresarse en forma general, de acuerdo al speech-act de la siguiente manera:

<mensaje>::=(<inf-act-stat> | <BD-act-stat> | <Asn-act-stat> | <Req-act-stat> |
<Mresp-act-stat> | <Def.cap-act-stat> | <Notif-act-stat> | <Conect-act-stat>
| <Fac-act-stat>)
< inf-act-stat> ::= Tell | Deny | Untell |
<BD-act-stat> ::= Insert | Delete | Delete_One | Delete_All
< Asn-act-stat> ::=Error | Sorry
<Req-act-stat> ::=Evaluate | Reply | Ask-if | Ask-about | Ask-one

| Ask-all
<Mresp-act-stat> ::= Stream-about | Stram-all | Eos
<Def.cap-act-stat> ::= Advertise
<Notif-act-stat> ::= Subscribe | Monitor
<Conect-act-stat> ::= Register | Unregister | Forward | Broadcast
| Pipe | Break | Transport_Adress
<Fac-act-stat> ::= Broker-one | Broker-all | Recommend-one
| Recommend-all | Recruit-one | Recruit-all

En la tabla 2 se muestra el conjunto de mensajes para KQML con su significado y función asociado a cada uno de ellos.

Conjunto de Mensajes de KQML

Nombre	Significado
Archieve	S quiere que R haga algo verdadero en su medio ambiente
Advertise	S esta particularmente disponible para procesar un "mensaje"
ask-about	S quiere todas las sentencias relevantes de la VKB[2] de R
ask-all	S quiere las respuestas de todos los R`s a una pregunta
ask-if	S quiere saber si la sentencia está en la VKB de los R´s
ask-one	S quiere una de las respuestas de los R`s a una pregunta
Break	S quiere que R rompa una línea establecida
Broadcast	S quiere que R envíe un "mensaje" a todas las conexiones
broker-all	S quiere que R colecte todas las respuestas a un mensaje
broker-one	S quiere que R obtenga ayuda en respuesta a un mensaje
Deny	El mensaje contenido no se aplica a S (nunca mas)
Delete	S quiere que R elimine una sentencia de su VKB
delete-all	S quiere que R elimine todas las sentencias que coincidan de su VKB
delete-one	S quiere que R elimine una sentencia que coincida de su VKB
Discard	S no quiere que los R`s mantengan respuestas a mensajes previos
Eos	Fin de una corriente de respuestas a una pregunta anterior
Error	S considera que el mensajes anterior de los R`s está mal formado
Evaluate	S quieres que R simplifique la sentencia
Forward	S quiere que R enrrute el mensaje
Generator	Lo mismo que un estandby de una corriente completa
Insert	S pide a R que agregue el contenido a su VKB
Monitor	S quiere actualizar las respuestas de los R`s a una corriente total
Next	S quiere la siguiente respuesta de R´s a un mensaje previo
Pipe	S quiere que R enrrute todos los mensajes a otro agente
R[2]eady	S esta listo para responder a los mensajes previos de R`s
Recommend-all	S quiere los nombres de todos los agentes que pueden responder un mensaje

[2] VKB Base Virtual de Conocimiento

Recommend-one	S quiere el nombre de un agente que pueden responder un mensaje
Recruit-all	S quiere que R obtenga todos los agentes disponibles para responder a un mensaje
Recruit-one	S quiere que R obtenga otro agente responda a un mensaje
Register	S puede entregar mensajes a un agentes designado
Reply	Comunica una respuesta esperada
Rest	S quiere que R´s proporcionen las respuestas a un mensaje previo
Sorry	S no puede proporcionar una respuesta mas detallada
Standby	S quiere que R este listo para responder a un mensaje
stream-about	Versión de Respuesta múltiple para ask-about
Stream-all	Versión de respuesta múltiple para ask-all
Subscribe	S quiere actualizar las respuestas de R`s a un mensaje
Tell	La sentencia está en la VKB de S`s
Transport-address	S asocia un nombre simbólico con una dirección de transporte
Unregister	Un deny de un register
Untell	La sentencia no está en la VKB de S`s

Tabla 2: Conjunto de mensajes Para el Emisor S y el receptor R

Como se ilustró en la fig.4.4 el LCIASA toma como base los mensajes de KQML para formar las sentencias compuestas que forman al lenguaje. La tabla 3 muestra los mensajes del lenguaje LCIASA con su función asociada.

Mensajes del lenguaje LCIASA

Admite-creencia	Agrega una creencia en la VKB del agente
Agrega-Plan	Incorpora un plan en la librería de planes
Analiza-plan	Determina que plan emplear para satisfacer una intención
Autentifica-agente	Verifica que el agente sea "amigo"
Autentifica-emisor	Comprueba que el emisor sea el esperado
Autentifica-mensaje	Certifica que el mensaje recivido es válido
Autentifica-Receptor	Compruba que el receptor sea el esperado
Confirma-firma-digital	Verifica que una firma digital sea válida
Compara-soluciones	Hace un análisis comparativo de las soluciones
Comprueba-plan	Confirma que el plan sea ejecutado correctamente
Dispone-mensaje	Hace que le mesaje sea observable
Elimina plan	Suprime un plan de la librería de planes
Excluye-VKB	Omite la VKB de las creencias del agente
Encuentra-agentes	Localiza a los agentes disponibles para ejecutar un mensaje
Encuentra-resuelve-mensaje	Busca agentes y obtiene la solución a un mensaje
Incorpora VKB	Agrega la VKB a las creencias del Agente
Localidades-de-Informacion	Indica las direcciones en donde se encuentra la información
Localiza-todas-soluciones	Proporciona todas las soluciones a un mensaje

Omite-creencia	Excluye la creencia de la VKB del agente
Protege al mensaje	Codifica al mensaje
Responde-con-la solución	Proporciona la información que se busca
Valida-Agente	Acredita al agente
Verifica-Orden-Mensajes	Confirma que la secuencia de mensajes sea consistente
Visuaiza VKB	Revela el contenido de la VKB

Tabla 3: Conjunto de mensajes para LCIASA

La sintaxis de LCIASA en notación BNF se expresa de la siguiente manera:

< LCIASA> :: = <sentencia>{ <sentencia>}*
<sentencia> ::= <acciones-compuestas>
<acciones-compuestas>::=<Admite-creencia> | <Agrega-plan> | <Analiza-plan>
| <Autentifica-agente> | <Autentifica-emisor> | <Autentifica-mensaje>
| <Autentifica-Receptor > | < Confirma-firma-digital> | < Compara-soluciones>
| <Comprueba-plan> | <Dispone-mensaje> | <Elimina-plan> | <Excluye-VKB>
| <Encuentra-agentes> | <Encuentra-resuelve-mensaje> | <Incorpora-VKB>
| < Localidades-de-Informacion> | < Localiza-todas-soluciones>
| <Omite-creencia> | < Protege al mensaje> | < Responde-con-la solución>
| < Valida-Agente> | < Verifica-Orden-Mensajes> | <Visualiza VKB>

En donde cada una de las sentencias de LCIASA (AOL) corresponde a una secuencia de mensajes de KQML (acciones del speech-act), es decir:

<acciones-compuestas>::= <mensaje>{ <mensaje>} *

Es posible representar las sentencias de LCIASA de manera general y de modo explícito, en siguiente forma:

< Admite-creencia> ::= <mensaje>{ <mensaje>} *
< Agrega-plan > ::= <mensaje>{ <mensaje>} *
< Analiza-plan >::= <mensaje>{ <mensaje>} *
< Autentifica-agente > ::= <mensaje>{ <mensaje>} *
< Autentifica-emisor >::= <mensaje>{ <mensaje>} *
<Autentifica-mensaje> ::= <mensaje>{ <mensaje>} *
<Autentifica-Receptor > ::= <mensaje>{ <mensaje>} *
< Confirma-firma-digital> ::= <mensaje>{ <mensaje>} *
< Compara-soluciones> ::= <mensaje>{ <mensaje>} *

<Comprueba-plan> ::= <mensaje>{ <mensaje>} *
<Dispone-mensaje> ::= <mensaje>{ <mensaje>} *
<Elimina-plan> ::= <mensaje>{ <mensaje>} *
<Excluye-VKB> ::= <mensaje>{ <mensaje>} *
<Encuentra-agentes> ::= <mensaje>{ <mensaje>} *
<Encuentra-resuelve-mensaje> ::= <mensaje>{ <mensaje>} *
<Incorpora-VKB> ::= <mensaje>{ <mensaje>} *
<Localidades-de-Informacion> ::= <mensaje>{ <mensaje>} *
<Localiza-todas-soluciones> ::= <mensaje>{ <mensaje>} *
<Omite-creencia> ::= <mensaje>{ <mensaje>} *
<Protege al mensaje > ::= <mensaje>{ <mensaje>} *
< Responde-con-la solución > ::= <mensaje>{ <mensaje>} *
<Valida-Agente> ::= <mensaje>{ <mensaje>} *
<Verifica-Orden-Mensajes> ::= <mensaje>{ <mensaje>} *
<Visualiza VKB> ::= <mensaje>{ <mensaje>} *

En donde los mensajes corresponden a las acciones primitivas de KQML. Definidas anteriormente en la tabla 2.

4.3.2 Semántica

La semántica relaciona al lenguaje con situaciones o con la representación de situaciones en el mundo. Con la semántica se obtiene una interpretación de las acciones ejecutadas durante el intercambio de información entre agentes.
La semántica del LCIASA está dada por un contexto en donde cada agente observa las capacidades de cada uno de los otros. Cada agente es visualizado como si administrara una base de conocimiento. Es decir, la comunicación con el agente se hace con respecto a su base de conocimiento.
La implementación del agente no es necesariamente estructurada como una base de conocimiento. Es decir se puede utilizar un simple sistema de bases de datos o un programa que utilice una estructura de datos especial, este programa debe trasladar el código empaquetado a una abstracción basada en conocimiento, para ser utilizada por otros agentes. Entonces se dice que cada agente maneja una base de conocimiento virtual (VKB).
Cuando se definen los "mensajes", es útil clasificar las sentencias dentro de la VKB en dos categorías: "creencias" e "intenciones". Una creencia de un agente codifica la información que se tiene del el mismo y de su medio ambiente,

incluyendo la VKB de otros agentes. Una intención codifica los estados de su medio ambiente externo que el agente tratará de alcanzar.
Los agentes sostienen una conversación alrededor de sus VKB de otras VKB utilizando el LCIASA, pero la codificación de las sentencias en la VKB puede utilizar una variedad de lenguajes de representación.

4.4 Protocolos de Interacción

El protocolo de Interacción se puede ver como una representación gráfica de la secuencia de mensajes intercambiados por los agentes para lograr un objetivo específico. Estas secuencias se agrupan para formar una sucesión coherente de acciones asociadas a un plan, intención o protocolo que el agente esté capacitado para ejecutar.
Los protocolos de Interacción derivados de las sentencias del lenguaje LCIASA son conceptuados como planes o secuencias de mensajes con el propósito asociado en la definición de las sentencias del lenguaje. La fig. 4.5 presenta los componentes considerados en el proceso de interacción entre agentes.

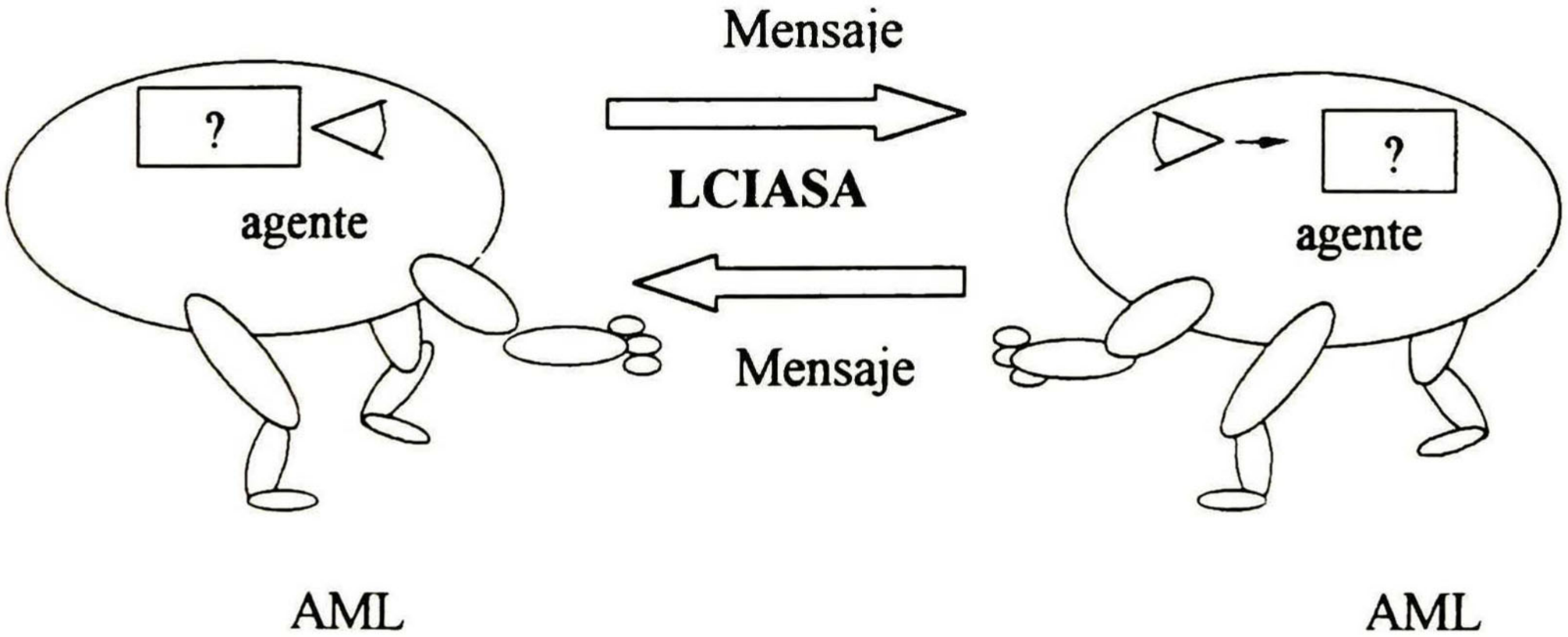


Fig. 4.5 Interacción entre agentes

En el proceso de interacción se logra el *entendimiento* entre los agentes por medio del intercambio de mensajes, a través de una negociación con la cual se alcanzan los objetivos deseados. Éstos objetivos son especificados como planes de acciones para alcanzar una meta específica. A continuación se ilustra la forma de definir los protocolos básicos y algunos protocolos de interés requeridos en el proceso de entendimiento entre los agentes.
La Fig. 4.6 muestra la secuencia de acciones que efectúa del agente A para encontrar al agente(en este caso el agente B) que este habilitado para responder a una solicitud del agente A y después obtener la información requerida de la VKB del agente B.

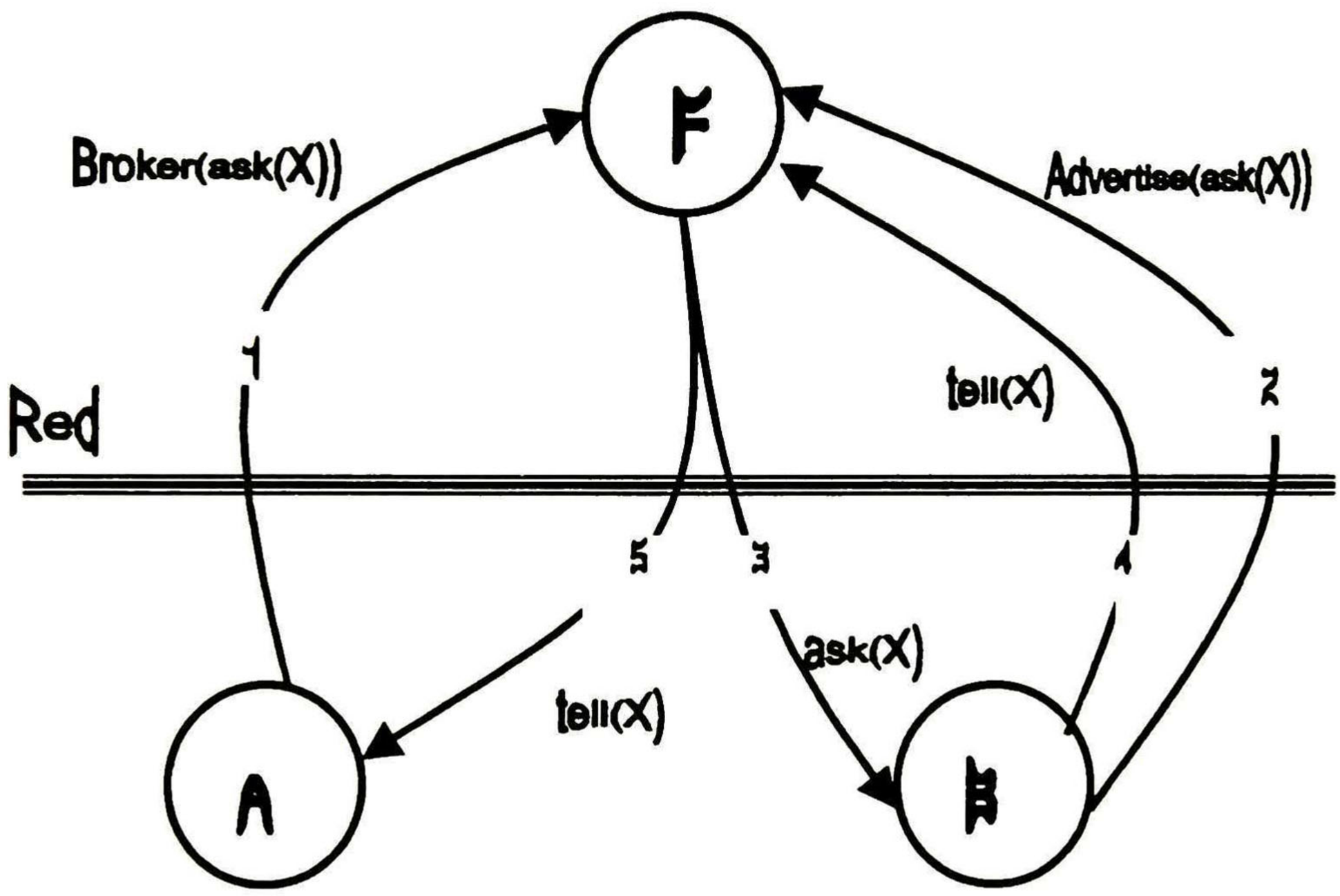


Fig 4.6: Protocolo Encuentra-Resuelve-Mensaje

En la Fig 4.6 se muestra la secuencia de acciones, las cuales corresponden a la sentencia de LCIASA Encuentra-Resuelve-Mensaje. Aquí se hace uso de un facilitador (F), el cual tiene la función de servir como una interfaz común entre los agentes.
De la misma manera a través del intercambio de mensajes se logra definir el conjunto de protocolos para LCIASA y otros que sean de interés a la aplicación que se realiza.

Ahora ilustramos el protocolo que nos permite comparar las soluciones a una consulta hecha por el agente A. La figura 4.7 representa el diagrama de

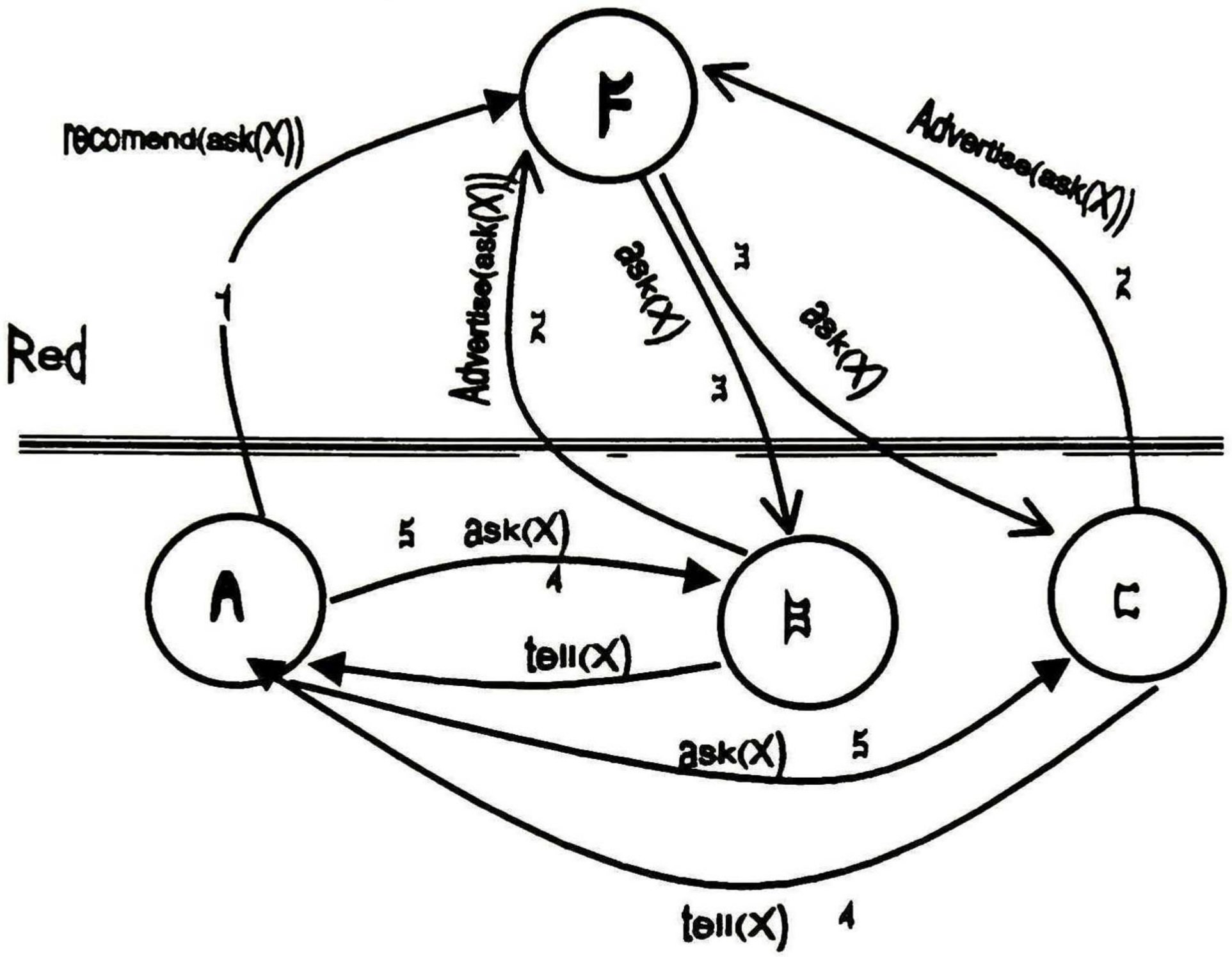


Fig. 4.7 Protocolo Compara soluciones

Transiciones. Éste muestra las acciones que realiza el agente A para encontrar a los agentes que estén habilitados para responder a una consulta hecha por este agente, una vez que se conocen los agentes dispuestos a efectuar la petición, el agente A sostiene un diálogo con los agentes dispuestos a responder con la solución, con este intercambio de mensajes el agente A recibe las respuestas de los agentes capaces, compara las soluciones e intercambia los mensajes correspondientes a la aplicación.

La figura 4.8 se muestra un esquema general para la especificación de protocolos. Éste esquema representa el mecanismo de interacción entre de los agentes para la especificación de protocolos.

La figura exhibe la posibilidad de especificar un Sistema MultiAgente y llevar a cabo sus funciones mediante un intercambio de mensajes, ésta secuencia ordenada de mensajes es conceptuado como un protocolo de interacción para llevar a cabo una función o tarea específica.

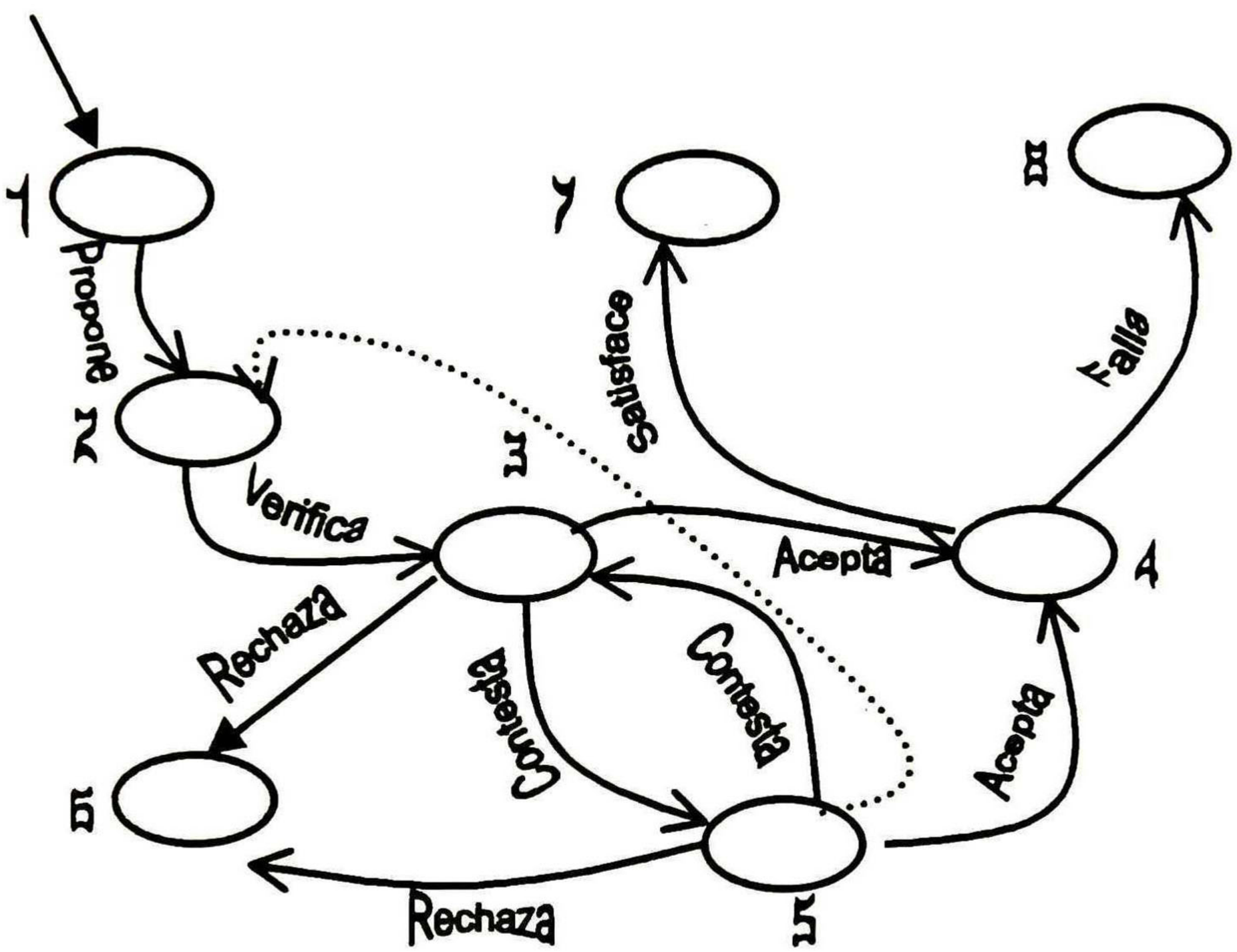


Fig. 4.8: Esquema general de transiciones para la interacción de agentes

Con el esquema general de transiciones para la interacción de agentes, es posible especificar todas y cada una de las sentencias de LCIASA como secuencias de las acciones primitivas de KQML, o como series de las sentencias del propio LCIASA.
A continuación se presentan las siguientes aplicaciones, que reflejan la utilidad y sencillez de del lenguaje LCIASA en la especificación de a Sistemas MultiAgente.

En la fig. 4.9 se presenta un esquema para un Sistema MultiAgente que permita la lectura/escritura de información en archivos compartidos. Aquí también se utiliza un facilitador (F) como una interfaz común para los agentes.

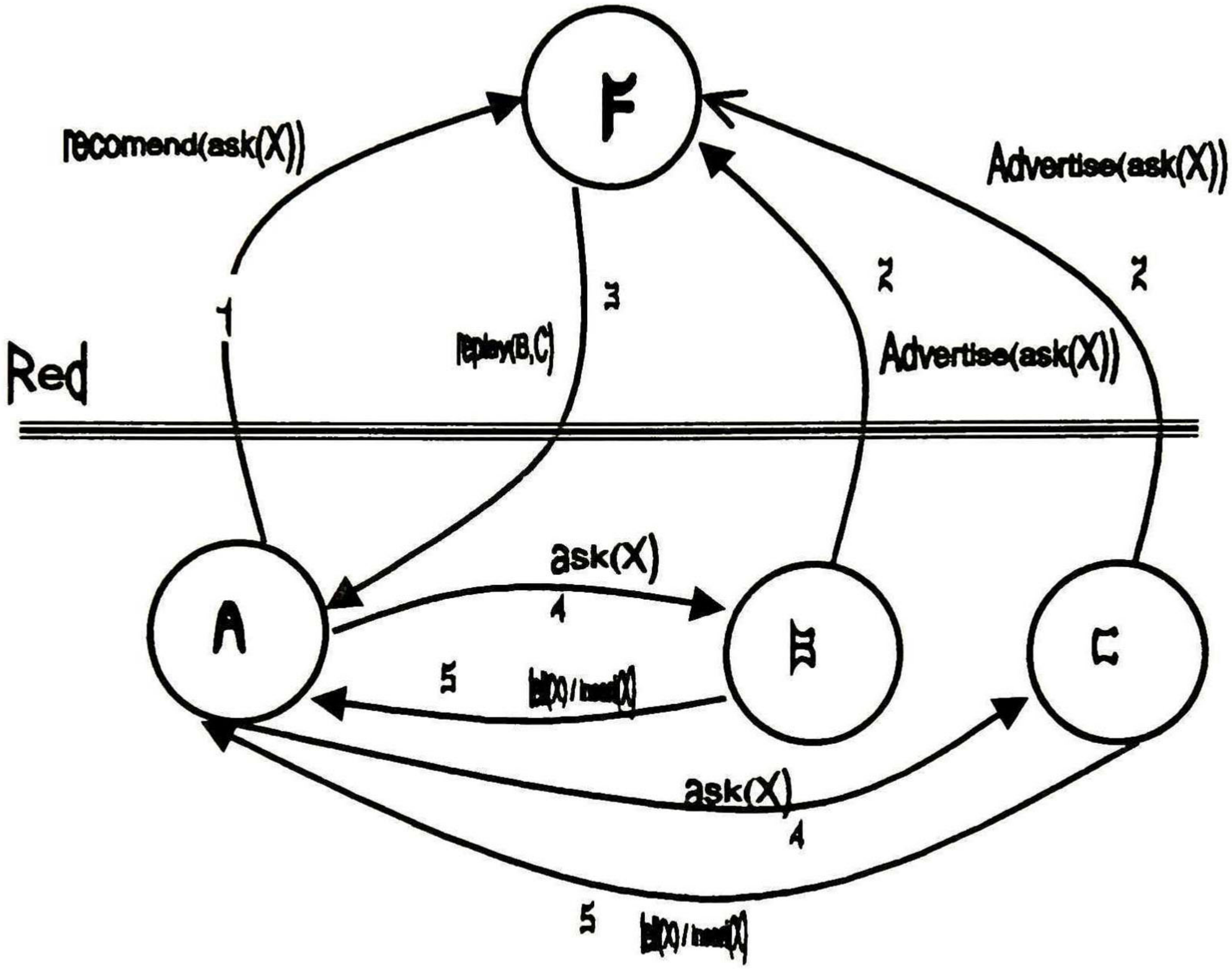


Fig. 4.9 Lectura/Escritura en archivos compartidos

En el paso 1 el agente A solicita a F los nombres de los agentes que estén posibilitados para procesar un mensaje, el facilitador cuestiona a los agentes para saber cuales pueden ejecutar el mensaje. En el paso2 los agentes A y B (es posible que sean mas) le notifican a F que están dispuestos a llevar a cabo la acción dada por el mensaje. En el paso 3 el facilitador le comunica al agente A que los agentes B y C están capacitados para ejecutar la acción del mensaje. En el paso 4 el agente A solicita a los agentes B y C la información solicitada en el mensaje y en el paso 5 los agentes Ay C envían al agente A la información requerida o escriben en la VKB del agente A dicha información.

Como otra aplicación de la especificación de Sistemas MultiAgente, la fig. 4.10 representa la conversación (Interacción) que sostienen los agentes para negociar la adquisición de un producto. La representación se hace por medio de una gráfica de Dooley [9], modificada por Parunak (1996).

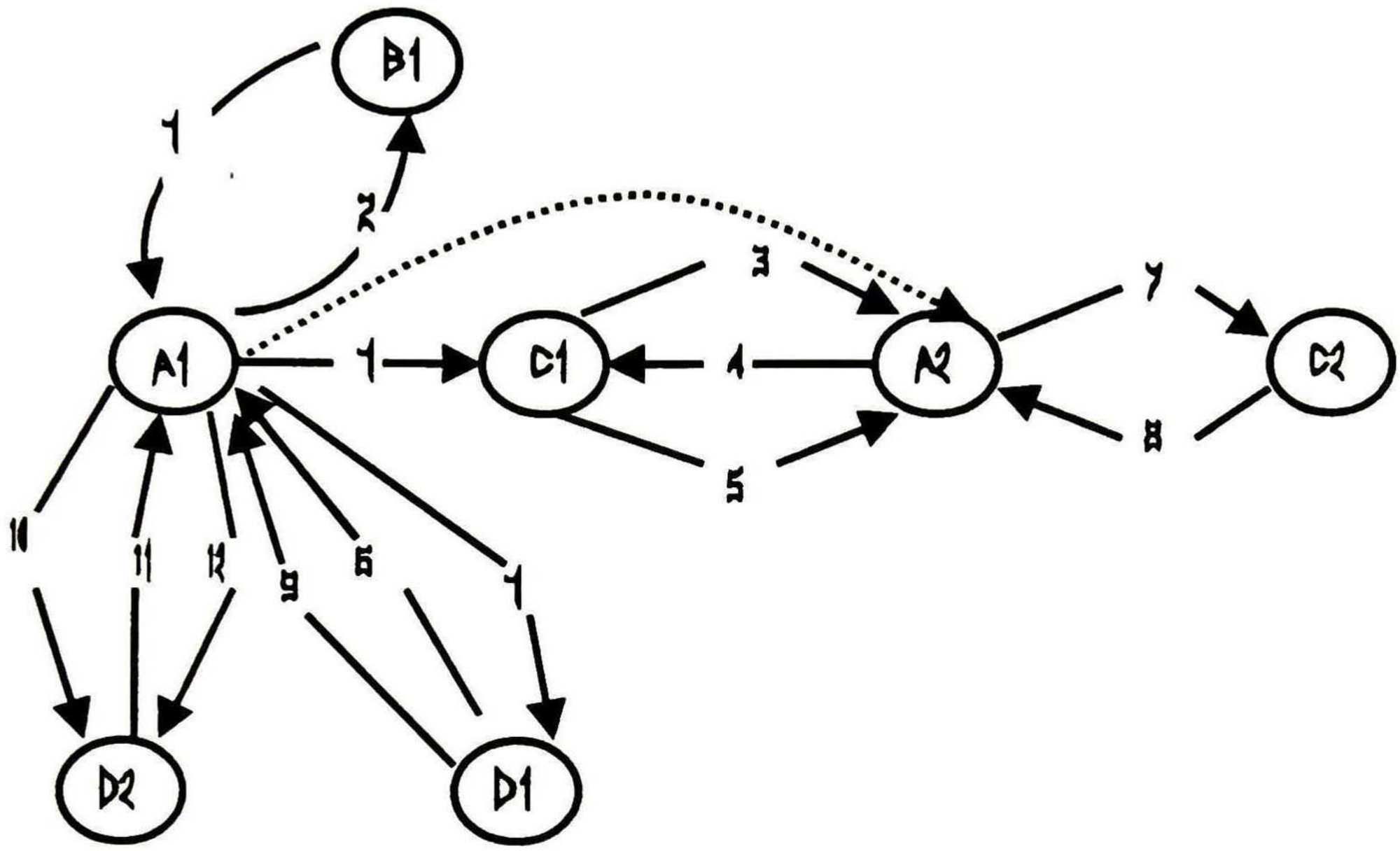


Fig. 4.10: Interacción para la adquisición de un producto

Las acciones de conversación de la fig. 4.10 se representan en la tabla 4 en donde se especifican las acciones realizadas, los agentes involucrados y una descripción de las actividades que se llevan a cabo.

Sec.	Emisor	Recept.	Descripcion
1	A	B,C,D	Encuentra-resuelve mensaje(ask(x)): Envíame 120 cajas de papel para impresora el próximo jueves.
2	B	A	Deny: No puedo hacerlo.
3	C	A	Tell: tengo 90 de papel para impresora al precio de catálogo para el viernes.
4	A	C	Replay: Favor de enviarme 90 cajas al precio de tu catálogo para el Viernes.
5	C	A	Replay: Correcto, te envío 90 cajas de papel para

			impresora el próximo Viernes.
6	D	A	Tell: Correcto, te envío 120 cajas de papel para impresora al precio de catalogo en próximo Jueves
7	A	C	Suscribe: Encontré un mejor proveedor, cancela el pedido.
8	C	A	Tell: Enterado, El pedido está cancelado
9	D	A	Tell: Confirma el envío
10	A	D	Replay: Solicitud de envío confirmada
11	D	A	Tell: Enterado, el pedido está registrado.

Tabla 4: Descripción de la Interacción entre agentes

Como se mencionó anteriormente LCIASA tiene asociado el metalenguaje AML, con el cual podemos hacer una especificación formal del SMA. Para ejemplificar esta asociación de lenguajes, a continuación empleamos AML para especificar la aplicación expuesta en la fig. 4.9 y por la tabla 4: La tabla 5 muestra la asociación de AML con LCIASA para especificar un SMA, con esta especificación es posible verificar la consistencia, obtener equivalencias y conocer los alcances y demás propiedades del sistema especificado.

Secuencia	Acciones
1	Send (a,b, Request(α) $\wedge$ (Bel b friend(a))
	Send (a,c, Request(α) $\wedge$ (Bel c friend(a))
	Send (a,d, Request(α) $\wedge$ (Bel d friend(a))
2	Send (b,a Abort(α))
3	Send (c,a, Request(tell(x))
4	Send (a,c, Request(replay(x))
5	Send (c,a, Inform(ϕ)) $\wedge$ (Bel a, ϕ)
6	Send (d,a, Inform(ϕ))
7	Send (a,c, Request(suscribe(x))
8	Send (c,a, Abort(α)) $\wedge$ (Bel a, ϕ)
9	Send (d,a, Request(tell(x))
10	Send (a,d, Request(replay(x))
11	Send (d,a, Request(tell(x)) $\wedge$ (Bel a friend(d))

Tabla 5: Especificación Formal de la Interacción de agentes

La ventaja de utilizar LCIASA en vez de KQML se da por la expresividad y naturalidad de las sentencias de LCIASA, es decir LCIASA es considerado como un dialecto de mayor nivel a KQML. Esto se puede mostrar en la aplicación anterior, en donde solo se requieren las sentencias:

Encuentra-resuelve–mensaje (Fig. 4.6)
Compara-soluciones (Fig. 4.7)

Para conseguir la misma información que se obtuvo en la aplicación expresada por la fig. 4.8 y con en la tabla 4. En esta práctica se observa que las sentencias de LCIASA nos permiten simplificar la especificación de las aplicaciones de Sistemas MultiAgente.
Además LCIASA tiene asociado el lenguaje AML para hacer la especificación formal del sistema propuesto. Como se muestra en la tabla 5. Esto le proporciona una mayor ventaja a LCIASA sobre los demás lenguajes de agentes de software.

Conclusión: Como se muestra en las aplicaciones el lenguaje LCIASA resulta apropiado para la especificación de SMA por la expresividad, legibilidad, confiabilidad y facilidad de sus sentencias desarrollar aplicaciones MultiAgente. Especialmente aplicaciones de comercio electrónico (compra-venta de artículos), transacciones bancarias, aplicaciones industriales.
Este capitulo presento el paradigma que genera al lenguaje LCIASA, La arquitectura del agente, con la cual se describe la funcionalidad y las características del agente. Así mismo se especifica el lenguaje LCIASA, su sintaxis, su semántica, el conjunto de sentencias del lenguaje y los protocolos asociados al lenguaje. Como punto final se presenta la aplicación de la especificación de Sistemas multiAgente en donde se muestra la asociación entre el metalenguaje formal AML y el lenguaje LCIASA con las cuales se remarca la importancia del lenguaje para lograr aplicaciones robustas en sistemas Distribuidos abiertos.

Capítulo 5

Conclusiones y Trabajo Futuro

En este capítulo se presenta un resumen de los resultados obtenidos, comentarios acerca del trabajo, las conclusiones obtenidas así como algunas sugerencias para continuar con el trabajo en un futuro.

5.1 Resultados Obtenidos

Como consecuencia de la realización del presente trabajo se mencionan entre otros los siguientes resultados obtenidos:

- Una metodología para implementar sistemas multiagente
- Un modelo formal para describir a los agentes con el cual es posible modelar sistemas Multiagente en forma genérica.
- En lenguaje LCIASA para especificar la interacción entre agente por medio de un conjunto de sentencias legibles a todo tipo de usuario. Con el cual es posible elaborar todo tipo de aplicaciones multiagente.

Con los resultados obtenidos se facilita el estudio de las aplicaciones actuales y futuras de los sistemas multiagentes, como lo son: aplicación de realidad virtual, comercio electrónico y en general sistemas de información distribuida.

5.2 Observaciones

El estudio de los sistemas multiagente, es un campo de gran importancia en todas las áreas del conocimiento, pero en especial en la Ingeniería, al cual se debe dedicar una gran cantidad de esfuerzos, dado que son conocidos los límites del paradigma cliente-servidor y objeto. Una de las principales motivaciones de este paradigma es la necesidad actual de manejar grandes volúmenes de información y de recursos físicos integrados en redes de computadoras.

Basado en resultados recientes en esta área, queda expuesta la importancia de los Sistemas Multiagente en la solución de estos problemas, sin embargo existes muchos problemas aún sin resolver.

5.3 Aportaciones del trabajo de tesis

- Especificación de un lenguaje que utiliza expresiones cotidianas par lograr el acceso a la información en sistemas distribuidos abiertos.
- Desarrollo de modelos formales que faciliten la utilización de herramientas computacionales (CORBA, JINI, JAVA, etc.) sujetas a estándares universales, flexibles y poderosos para el desarrollo de aplicaciones distribuidas robustas en plataformas heterogéneas.
- Se presenta un panorama completo del campo de los sistemas mutiagente, abarcando la evolución de los tres principales elementos de estudio: su arquitectura, Modelos teóricos y los lenguajes de agentes.
- Propuesta de un enfoque metodológico para el estudio de la interacción en Sistemas MultiAgentes
- Impulsar el desarrollo de aplicaciones reales (en la industria, comercio, educación, entretenimiento, etc.) que conduzcan a reemplazar los poco eficientes sistemas centralizados por sistemas distribuidos.
- Facilitar el desarrollo de Sistemas MultiAgente acorde al desarrollo actual de las tecnologías de comunicación (Comercio electrónico, realidad virtual, etc.).
- Adaptación a los nuevos modos de procesamiento de la información.

5.4 Conclusiones

El LCIASA se presenta como una herramienta para enfrentar los cambios que ha sufrido la tecnología del procesamiento de la información en la última década (aplicaciones de comercio electrónico, realidad virtual, Interfaces Hombre - Máquina, etc.). La instrumentación del tema de investigación permite generar una herramienta tecnológica que esté sujeta a estándares internacionales, además de ser una herramienta flexible, genérica de aplicaciones orientadas hacia los sistemas distribuidos. Esta herramienta es

apropiada para desarrollar múltiples de las aplicaciones actuales, asimismo es adecuada para aplicaciones actuales y futuras.
La asociación de LCIASA con el Metalenguaje de Agentes AML ofrece una perspectiva innovadora para la implementación de Sistemas MultiAgente, la razón es que AML, se presenta como una herramienta de modelado y diseño de Sistemas MultiAgente (SMA). Con esta herramienta es posible comprobar la consistencia de una especificación para un SMA, así mismo es posible estudiar el comportamiento, características y capacidades del sistema especificado.
Como punto final se menciona que la tecnología de agentes es el camino correcto, para resolver los problemas complejos que demandan los sistemas de procesamiento de información, en nuestros días y en lo venidero.

5.5 Trabajo Futuro

El lenguaje LCIASA especifica el lenguaje para la interacción entre agentes, en el que se define la arquitectura, el paradigma, la sintaxis, la semántica y los protocolos asociados que lo definen. Estos elementos proporcionan la base para un trabajo futuro en el que se implemente un esquema formal de cooperación entre agentes basado en planes.
Otras opciones de trabajos a desarrollar en un futuro son:

- Implementar LCIASA utilizando como plataforma de desarrollo a Java, CORBA ó Tcl &Tk.
- Asimismo esta la posibilidad de desarrollar un modelo formal completo con el cual se pueda especificar para la interacción de sistemas multiagente. Tomando como apoyo este modelo es posible trabajar en un futuro desarrollando un sistema de cómputo que opere como una máquina de estados, con la cual sea posible especificar y verificar un Sistemas Multiagente en forma automática.
- De la misma forma podemos utilizar el modelo formal mencionado anteriormente para aplicarlo en la verificación de diversos sistemas industriales y comerciales.
- Utilizar LCIASA para probar la equivalencia de Sistemas MultiAgente

Referencias

[1] Barthes, J.P Multi-Agent Systems, International Symposium on advanced Distributed Systems, Guadalajara, Jal. Marzo de 2000

[2] Appleby, S. & Steward, S. "Mobile Software Agents for Control in telecommunications Networks," BT Techonological Journal 12 (2) pp. 104-113, April 1994.

[3] Chaib-draa, B. Moulin, B. Mandiau, R. & Millot, P., " Chapter 1 – Trends in Distributed Artificial Intelligence, " *Foundations Of Distributed Artificial Intelligence*., G.M.P. O`Hare and N.R. Jennigs (eds) Jhon Wiley & SonsInc., pp 3-55 1996.

[4] Cohen, P.R. & Levesque, H.J., " Communicative Actions for Artificial Agents", ICMAS-95: *Proceedings, First International Conference* of MultiAgent Systems, pp 67-72, 1995.

[5] Cybele Agent Infrastructure, http://www.i-a-com/projects/cybele/index.html., 1996

[6] Connah, D., " The Design of Interacting Agents for Use in Interfaces", *Human-Machine Communication for Educational Systems Design, NATO ASI Series F, Computer and Systems Sciences* 129, Brower-Janse, D. & Harringdon,T.L. (eds.), Heidelberg: Spring Verlang, pp. 79-95, 1994.

[7] Dimitris N. Chorafas, Agent Technology Handbook, Mc Graw Hill, 1998

[8] Dooley,R.A. "Appendix B-Repartee as agraph," *An anatomy of Speech Notions*, pp 384-358, 1976.

[9] Dufee, E.H.,Lesser, V.R. & Corkill,D.,"*Coherent Cooperation among Communicating Problem Solvers*," *IEEE Transactions on Computers* C-36(11), pp 1275-1291, 1987.

[10] Fisher, K., Muller, J.P & Pischel, M., " Unifying Control in a Layered Agent Architecture", *Technical Report TM-94-05*, German Research Center for AI-(DFKI GmnH), 1996.

[11] Gasser, L., Rosenchein, J.S. & Ephrati, E., "Introduction to Multiagent Systems". *Tutorial Presented at the 1st International Conference* on Multi-Agent Systems, San Francisco CA., June 1995.

[12] Genesereth, M. R. & Ketchpel, S. P., "Software Agents", Communications of the ACM 37 (7), pp. 48-53, 1994.

[13] G.M.P O´ Hare, and N.R. Jennings, *Fondations of Distributed Intelligence*, Jhon Wiley & Sons Inc.

[14] Georgeff, M., "Agents with Motivation : Essential Technology for Real World Applications", *The First International Conference of the Practical Applications of Intelligent Agents and Multi-agent Technology*, London, UK, 24th April 1996.

[15] Heilmann, K., Kihanya D. Light A., & Musembwa, P., " Intelligent Agents a Technology and Business Application analysis" http://haas.berkely.edu./~heilmann/, 1995

[16] Huhns, M.N. & Sing, M.P., " Distributed Artificial Intelligence for Information Sistems, "CKBS-94 Tutorial, University of Keele, UK, June 15, 1994.

[17] Jennigs, N.R., " Specification and Implementation of a Belief Desire Joint Intention Architecture for Colaborative Problem Solving," *Journal of Intelligent and Cooperative Information Systems* 2 (3), pp. 289-318, 1993

[18] Lynch Nancy A., *Distributed Algoritms*, Morgan Kaufmann Inc. San Francisaco CA., 1996.

[19] Maes, P (ed), *Designing Autonomous Agents Theory and Practice from Biology to Engineering and Back*, London, The MIT Press, 1991.

[20] Michael Wooldidge, Jorg P Muller, Milind Tambe, *Intelligent Agents, Lecture Notes in Artificial Intelligence*, Vol I,II, III, Springer Verlang.

[21] Nwana.H.S. & Wooldridge, M., "Software Agent Technologies", British Telecommunications Technology Journal 14 (4),pp.167-27, October 1996.

[22] Parunak,H.Van Dyke,"Chapter 4- Applications o Distributed Artificial Intelligence in Industry, "*Foundations of Distributed Arificial Intelligence*, G. M. P O'Hare and R. Jennings (eds.),John Wiley &sons, pp. 139-163, 1996.

[23] Ramos Corchado, F.F., " Fundamentos de Inteligencia Artificial Distribuida" *Escuela de Invierno*, Guadalajara Jal. 1998

[24] R. De Nicola et al, KLAIM: "*A Kernel Languaje for agents Interaction and mobility*" , IEEE Transactions on SOFTWARE ENGINEERING, pp. 315-330, May 1998.

[25] Rao, A.S. & Georgeff, M.P., "BDI Agents From Theory to Practice" In Preceedigs of the 1st International Conference on Multi-Agent Systems(ICMAS-95) San Francisco CA., USA, June, pp 312-319, 1995.

[26] Russell Norvig, *Artificial Intelligence: a Modern Approach* , Prentice Hall., 1996.

[27] Sape Mullender, *Distributed Systems.*, Addison Wesley, 1993.

[28] Tim Finn, Don Mc Kay, Rich Fritzon, *An Overview of KQML. A Knowlwdge Query and Manipulation Languaje*, University o Maryland, Baltimore MD 21228.

[29] Torres González, R.E., "Lógica Modal : Apuntes del curso de LOGICA", *Cinvestav-IPN, Unidad Académica Guadalajara, Jal*, 1998.

[30] Torres González, R.E., "Lógica Temporal · Apuntes del curso de LOGICA", *Cinvestav-IPN, Unidad Académica Guadalajara, Jal*, 1998.

[31] Wooldridge, M., *Conceptualizing and Developing Agents*, Proceedings of the UNICOM Seminar on Agent Software, 25-26 April, London, pp. 40-54, 1995.

[32] Wooldridge, M., " Chapter 10 –Temporal Belief logic for modelingDistributed Artificial Intelligence systems, " *Foundations Of Distributed Artificial Intelligence.*, G.M.P O`Hare and N.R. Jennigs (eds) Jhon Wiley & SonsInc., pp 269-285 1996.

Apéndice A Glosario

ACL Lenguaje de comunicación entre agentes

ACTOR Concepto pionero de programación concurrente

AGENTE En su uso mas general se conceptúa como un sistema de cómputo autónomo, reactivo, pro-activo y auto–contenido. También es considerado en términos de conceptos más humanos como las creencias, deseos e intenciones.

AGENTE CREIBLE Es un agente que generalmente es representado en un juego de computadora o en alguna clase de ambiente virtual.

AGENTE DE INTERFACE es un programa de computadora que emplea técnicas de Inteligencia Artificial para proporcionar asistencia a un usuario que esta tratando con alguna aplicación

AGENTE DE SOFTWARE No es un agentes implementado por software. Más bien es un Agente que opera como sensor y actúa en un ambiente de software tal como UNIX.

ARQUITECTURA Metodología particular para la construcción de agentes, típicamente incluye definiciones de software, estructuras de datos y operaciones sobre las estructuras.

ARQUITECTURA BDI Es una arquitectura que contiene una representación explícita de creencias, deseos e intenciones. Las creencias son consideradas como la información que el agente tiene de su medio ambiente, las cuales pueden ser falsas. Los deseos son todas aquellas cosas que el agente quisiera realizar, los deseos no son todos consistentes, y no esperamos que el agente actúe sobre todos ellos. Las intenciones son aquellas cosas a las que el agente se compromete a realizar.

ARQUITECTURA DELIBERATIVA Arquitectura que se basa en una representación interna dada por un modelo simbólico y una manipulación simbólica.

ARQUITECTURA HIBRIDA Arquitectura que trata de cazar las técnicas de la Inteligencia Artificial simbólica con otras aproximaciones para agentes.

Apéndice A Glosario

ARQUITECTURA REACTIVA Arquitectura que no emplea clase alguna de modelo simbólico central del mundo.

AGENTE CREIBLE Es un agente que generalmente es representado en un juego de computadora o en alguna clase de ambiente virtual.

ARQUITECTURA DE PIZARRON Es una clase de arquitectura en la cual una colección de fuentes de conocimiento se comunican escribiendo en una estructura de datos accesible en forma global, conocida como pizarrón.
AUTONOMIA El hecho de asumir que el agente nunca actúa sin la intervención humana directa u otra intervención y que tienen alguna clase de control sobre su estado interno.

CONATIVO Hacer con deseo

COORDINACION Proceso para asegurar que las partes de un problema sean incluidas en al menos un agente, para completar la solución de un problema.

ESTADO COGNITIVO Estado interno de un sistema Intencional. la colección de creencias, deseos e intenciones, que caracterizan a un agente en un instante dado.

ESTADO MENTAL estado Cognitivo.

EPISTEMICO Hacer con conocimiento

FACILITADOR Interfaz Común para los agentes

LENGUAJE DE AGENTE Lenguaje de programación que considera la noción de agente, ejemplo telescript, Agente 0.

LOGICA MODAL Lógica de necesidad y posibilidad, las técnicas de la lógica modal se han usado para formalizar nociones mentales así como aspectos temporales.

NEGOCIACION Es el proceso para alcanzar un estado que es mutuamente conveniente a un conjunto de agentes, esta íntimamente relacionado con la coordinación.

ONTOLOGIA Entidad computacional, un recurso conteniendo conocimiento acerca de que "conceptos" existen en el mundo y como estos se relacionan con otros

PLAN es un representación de un curso de acciones , que cuando son ejecutadas conducen a alcanzar un objetivo, los planes pueden involucrar a varios agentes.

Apéndice A Glosario

PRO –ACTIVO Capaz de tomar la iniciativa. no solamente manejado por eventros, capaz de generar metas y actuar racionalmente para lograrlas.

PROGRAMACION ORIENTADA-AGENTES Una aproximación para la construcción de agentes la cual propone la programación de agentes en términos de nociones mentales, como deseo, creencia e intención.

PROTOCOLO DE COOPERACION Es un protocolo que define como un grupo de agentes trabajan juntos para alcanzar un objetivo. El mas conocido es el contract net.

SISTEMA INTENCIONAL Sistema cuyo comportamiento se puede predecir o explicar atribuyendo al sistema actitudes como creencias, deseos e intenciones, junto con un grado de racionalidad.

TEORIA DEL SPEECH-ACT Teoría pragmática de la comunicación. Los axiomas principales de la teoria del speech-act se basan el hecho de que las expresiones comunicativas son acciones ejecutadas por un emisor y con la finalidad de iniciar un cambio en el estado mental de algún receptor.

VERACIDAD Asume que un agente no reconoce información falsa.

Apéndice B Notación

A El agente A
Ag Conjunto de agentes
I_A estados de información de los agentes
S_A estados estratégicos o intencionales de A
V_A estado de evaluación para el agente A
◊ " Es posible que" o posiblemente
□ " es necesario que" o necesariamente
P□ " fue necesario que"
∀ para todo
∃ Existe
¬ Negación (not)
∧ Conjunción (and)
∨ Disyunción (or)
$\beta^o \in BS$ conjunto de creencias iniciales del agente
$A : BS \rightarrow Ac$ función de Acción del agente
$M : BS \rightarrow \wp(Mess)$ función de generación de mensajes del agente
$\eta : BS \times Ac \times \wp(Mess) \rightarrow BS$ función del siguiente estado del Agente.
(Bel $i\ \phi$) El agente i cree ϕ
(Send $i\ j\ \phi$) El agente i envió al agente j el mensaje ϕ
(Do i α) El agente i ejecuta la acción α
Oϕ El siguiente ϕ
⊗ϕ El último ϕ
$\phi\ \mu\ \psi$ ϕ hasta que μ (no estricto)
$\phi\ \omega\ \psi$ ϕ a menos que ψ
$\upsilon\phi$ ϕ hasta ahora
♦ϕ ϕ alguna vez
$\alpha\phi$ ϕ antes
$\phi\beta\psi$ ϕ Cuando ψ
$\phi\ \xi\psi$ ϕ desde que ψ

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: "LCIASA: Lenguaje de "Capacidad de Interacción" de Agentes en Sistemas Abiertos" del Sr. Nicandro Farías Mendoza, el día 8 de Septiembre de 2000.

EL JURADO

Dr. Luis Ernesto López Mellado
Investigador Cinvestav 3 A
CINVESTAV DEL IPN
Guadalajara.

Dr. Félix Francisco Ramos Corchado
Investigador Cinvestav 2 A
CINVESTAV DEL IPN
Guadalajara.

Dr. Victor Hugo Zaldívar Carrillo
Coordinador de la Maestría en Informática aplicada
Departamento de Electrónica, Sistemas e Informática
ITESO, Guadalajara.