Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

UTILIZACIÓN DE AGENTES AUTÓNOMOS EN ENTORNOS EMPRESARIALES

Alvaro Esteban Méndez

Asesorado por el Ing. Rodolfo Estuardo Arriaga Herrera

Guatemala, abril de 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

UTILIZACIÓN DE AGENTES AUTÓNOMOS EN ENTORNOS EMPRESARIALES

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

ALVARO ESTEBAN MÉNDEZ
ASESORADO POR EL ING. RODOLFO ESTUARDO ARRIAGA HERRERA

AL CONFERÍRSELE EL TÍTULO DE
INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, ABRIL DE 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Inga. Glenda Patricia García Soria
VOCAL II	Inga. Alba Maritza Guerrero de López
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Inga. Floriza Felipa Avila Pezquera
EXAMINADOR	Inga. Virginia Victoria Tala Ayerdi
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

UTILIZACIÓN DE AGENTES AUTONOMOS EN ENTORNOS EMPRESARIALES,

tema que me fuera asignado por la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería, con fecha enero de 2006.

Alvaro Esteban Méndez

AGRADECIMIENTOS A:

DIOS Por llevarme siempre de la mano y estar conmigo en mis momentos de tribulación, por darme retos en la vida y la fortaleza para salir adelante.

MI MADRE Por su apoyo, dedicación y fuerza que siempre han sido un ejemplo a seguir.

MI PADRE Por su apoyo, sabiduría, y por ser un ejemplo de perseverancia.

MIS HERMANOS Por su cariño, apoyo y por ser mi fuente de inspiración para salir adelante.

MI FAMILIA Porque los considero un pilar fundamental en mi vida y el legado más valioso que una persona pueda desear.

MIS AMIGOS Bruce, Charly, Chema, Carlos, Julio, Gerardo, Milton, Maco con quienes compartí gran parte de la carrera y entre todos logramos alcanzar cada una de las metas que nos propusimos.

MI ASESOR Ing. Rodolfo Estuardo Arriaga Herrera, por su orientación a lo largo del desarrollo del presente trabajo y por depositar su confianza en mí para la realización del mismo.

MIS AMIGOS Quienes compartieron conmigo durante mi formación y quienes siempre han sido una fuente inagotable de aprendizaje.

Y a todos aquellos que me ayudaron a completar mi formación profesional.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
AAS	Agente Administrador de la Sucursal
ACI	Agente Controlador de Información
ADP	Agente de Documentación de Préstamos
AFSM	Lenguaje de subsunción basado en una máquina de estados finito aumentada
AIM	Módulo de agente de gestión de información
AOP	Programación Orientada a Agentes
APOP	Agente de Prestamos de la Oficina Principal
ARC	Agente de informe de crédito
BDI	Familia de lógicas para especificar sistemas multiagente, creencia, deseo e intención
BEL	Descripción de operadores de creencias
CA/NA	Sistemas distribuidos completamente precisos
CBB	Comportamiento del comprador consumidor
CNP	El protocolo de contratos
CSP	Proceso de Comunicación Secuencial
DAI	Inteligencia Artificial Distribuida
DAML	Lenguaje para modelar ontologías
DES	Descripción de operadores de deseos
DPS	Resolución cooperativa de problemas distribuidos
EbXML	Electronic Bussiness XML, infraestructura abierta basada en XML, para negocios electrónicos
FA/C	Cooperación Funcionalmente precisa
FIPA	Fundación para agentes físicos inteligentes

GUI	Interfaz grafica de usuarios
HLCM	Módulo de comunicación de alto nivel
HTML	Lenguaje de marcas de hipertexto
IA	Inteligencia Artificial
INTEND	Descripción de operadores de intenciones
IS	Sistema Inteligente
JADE	Entorno de desarrollo de agentes en JAVA
KIF	Formato de intercambio del conocimiento
KQML	Lenguaje de consulta y manipulación del conocimiento
KS	Fuentes de conocimiento
MAS	Sistemas multiagente
MBD	Administracion por delegación
MHEG	Estándar de middleware de los servicios digitales
OIL	Lenguaje de Intercambio de Ontologías
OS	Sistema operativo
PCM	módulo de planeación y coordinación
PGP	Algoritmo de Planificación parcial Global
PI	Protocolo de interacción
PoBs	Conjunto de patrones de comportamiento
QUERY	Consulta
RDF	Marco de descripción de recursos
SAC	Servicio de Atención al Cliente
SCP	Puntos de control de servicio
UDDI	Descubrimiento e intregracion del descriptor universal
URL	Localización de recurso uniforme
WAN	Red de área ancha
WBI	Buscador web inteligente
WWW	Red de ancho mundial
XML	Lenguaje de etiquetado extensible

GLOSARIO

Agente Un agente es una entidad que reside en un entorno, donde interpreta datos de censores reflejando eventos del entorno, y ejecuta comandos que provocan cambios en éste.

Agente autónomo Un agente es autónomo en la medida en que su comportamiento está determinado por su propia experiencia y no en base a conocimiento predefinido por el diseñador.

Agentes de interfaz Tienen como objetivo simplificar las tareas rutinarias que realiza un usuario.

Agentes de Internet Tiene como objetivo navegar por la red recolectando información, indexarla y ofrecer la información que puede interesar al usuario cuando realiza una consulta.

Agentes de red Agentes que pueden acceder no solo a recursos locales sino también a los remotos, y tienen un conocimiento de la infraestructura de la red y de los servicios disponibles dentro de la red.

Agentes de software reactivos Agentes que no poseen modelos simbólicos internos de sus entornos, en su lugar reacciona o responde en modo de estímulo respuesta para representar el estado del entorno en el que están empotrados.

Agentes federados	Agentes que no se comunican directamente entre sí, sino que lo hacen a través de un facilitador o mediador que ubican a otros agentes en la red, los cuales son capaces de proporcionar otros servicios.
Agentes heterogéneos	Son los que integran diferentes clases de agentes y pueden además contener una o más clases de agentes híbridos.
Agentes híbridos	Agentes cuya constitución es una combinación de dos o más filosofías de agentes para formar un agente único.
Agentes locales	Agentes que acceden solamente a recursos locales.
CBB	Comportamiento del comprador consumidor, es la fase de la negociación donde el precio u otras condiciones de la transacción son determinados.
CNP	Protocolo de alto nivel que facilita el control distribuido de la ejecución de tareas de forma cooperativa.
Deadlocks	Las cerraduras muertas se dan cuando un agente sale de una negociación y la bloquea.
Ecosistemas	Forma de evaluar la dinámica global de los sistemas distribuidos.
FIPA	Fundación para agentes físicos inteligentes, organización internacional dedicada a promover la industria de agentes inteligentes.

JADE	Entorno de desarrollo de agentes para JAVA.
KQML	Lenguaje de consulta y manipulación del conocimiento
KS	Las fuentes de conocimiento KS son módulos especializados que intercambian mensajes a través de la pizarra.
Livelocks	Las cerraduras vivas se dan cuando un agente dentro de una negociación se bloquea por no dar seguimiento dentro del tiempo establecido de respuesta.
Ontología	Es una especificación explícita y formal sobre una conceptualización compartida.
Pizarra	La pizarra es una base de datos global compartida por todas las fuentes de conocimiento que contiene datos e hipótesis.
RDF	Identifica los denominadores comunes y proveer un mecanismo para que los diseñadores de páginas puedan proveer metadatos útiles para todos.
Sistemas Multiagente	Estudia la coordinación de la conducta inteligente entre un conjunto de agentes inteligentes autónomos.
Web semántica	Información que aparece en Internet que puede interpretarse por los ordenadores sin necesidad de intervención humana.

RESUMEN

La creciente expansión de los negocios, la incursión del Internet en casi todas las áreas comerciales, y el continuo flujo de información, han hecho que cada vez sean más necesarios los medios para manipular dicha información, sin embargo, y en gran medida a su rápido crecimiento, también, se ha convertido en uno de los medios más complicado de manipular a todas las escalas. Es por ello que la creación de sistemas que manejen la información se ha vuelto uno de los puntos cruciales en la infraestructura tanto de sitios en Internet, como de empresas, cuya información es un componente esencial.

Una estrategia de acercamiento de la tecnología a los usuarios, se encuentra en el desarrollo que se ha realizado de agentes autónomos, este interés se ha originado gracias a la convergencia progresiva de la informática y las telecomunicaciones o lo que se conoce como telemática.

Un agente autónomo es un sistema anidado y parte integrante de un ambiente, y que detecta o percibe datos ambientales, momento a momento, y actúa sobre él, a través del tiempo, persiguiendo sus propios objetivos de forma que afecte lo que siente en el futuro.

Los agentes son fragmentos de software con características humanas que facilitan el aprendizaje. Las características pueden expresarse desplegando texto, gráfico, iconos, voz, animación, multimedia o realidad virtual.

Los agentes habitan en entornos dinámicos complejos, perciben y actúan de forma autónoma en ese entorno, realizando un conjunto de tareas y cumpliendo objetivos para los cuales fueron diseñados.

En la web semántica serán los encargados de realizar la búsqueda de servicios, para ello, la semántica facultará a los agentes para describir unos a otros la función exacta que realizan, y qué datos han de recibir para ello. Pretenden ser programas avanzados que pueden funcionar útilmente en entornos de importancia para los seres humanos.

Por otro lado, las tecnologías más novedosas están en el campo del procesamiento y presentación de la información. Y no existe la información si el agente no es capaz de entenderla, por eso se utilizan estándares para el intercambio de información, que permitan que agentes creados por diferentes autores hablen el mismo lenguaje. Pero para que esa información afecte al estado interno y a las acciones del agente en el futuro, el agente debe de ser capaz de adaptarse.

La idea de este trabajo es crear una fuente de entendimiento sobre la utilización de agentes autónomos y las ventajas que pueden recibir las pequeñas y medianas empresas que decidan implementar este tipo de tecnología.

OBJETIVOS

General

Comprender la utilización de agentes, identificando la metodología de desarrollo, los componentes necesarios para la implementación de agentes en sistemas de información y evaluar los beneficios de su utilización en empresas de servicios y comercio electrónico.

Específicos

1. Identificar la metodología utilizada por los agentes autónomos para el logro de los objetivos trazados, permitiendo establecer los escenarios para optimizar el uso de los recursos, por medio de la identificación de las tendencias del entorno.

2. Documentar el proceso de construcción de agentes autónomos.

3. Identificar los tipos de modelos que se requieren para realizar un agente autónomo.

4. Desarrollar modelos necesarios para la construcción de agentes capaces de satisfacer las demandas de acuerdo al comportamiento del mercado.

5. Construcción de los modelos para la realización de un agente autónomo.

6. Evaluar beneficios al utilizar agentes autónomos.

INTRODUCCIÓN

La tendencia actual de los sistemas de comunicación es lograr una conectividad global e incrementar las opciones de comunicación disponibles para el acceso e intercambio de la información lo que indica que la visión para el futuro de las comunicaciones es información en cualquier momento, lugar y forma basados en los servicios electrónicos abiertos donde se ofrecerá un espectro ilimitado de servicios de comunicación e información. En este contexto el término agente ha venido sonando desde hace algunos años y su interés se ha originado gracias a la convergencia progresiva de la informática y las telecomunicaciones o lo que se conoce como telemática.

Básicamente la idea de agentes autonomos y las tecnologías han sido influenciadas por la gran variedad de disciplinas y prácticas. Sin embargo, el origen de este término es el campo de la inteligencia artificial en particular el de la inteligencia artificial distribuida. Se comenzó a usar a comienzos de los 80's para reflejar la idea de crear objetos que piensan por ejemplo entidades de software autónomas que están de acuerdo con una inteligencia autocontenida.

Con este trabajo se pretende dar una caracterización a los agentes que permita llegar a definir su tipología y ofrecer lo que sería la taxonomía y arquitectura de los mismos, además, sus campos de aplicación, evaluando los beneficios de su utilización.

1. FUNDAMENTOS DE LOS AGENTES INTELIGENTES

La tecnología de agentes es suficientemente madura para su aplicación masiva, y están surgiendo productos basados en esta tecnología para realizar funciones concretas. Sin embargo, la aplicación de esta tecnología en la industria necesita el desarrollo de métodos de análisis y diseño prácticos, que están comenzando a surgir.

1.1. Agentes como parte de la evolución de la inteligencia artificial[1]

La inteligencia artificial distribuida se ha definido como un subcampo de la inteligencia artificial que se centra en los comportamientos inteligentes colectivos que son producto de la cooperación de diversas entidades denominadas agentes. A pesar de ser una disciplina joven, podemos distinguir tres periodos cronológicos bien diferenciados:

- La inteligencia artificial distribuida "clásica" se centra en el estudio de la conducta colectiva, en oposición a la inteligencia artificial, que estudia la conducta individual
- La inteligencia artificial distribuida "autónoma" se centra en el estudio de los agentes individuales situados en un mundo social. Se modifica la visión inicial de la Inteligencia artificial distribuida en que la "perspectiva social" significa que la sociedad prima sobre los individuos, que se explican a partir de su función en la sociedad, y se centra en estudiar agentes autónomos en un mundo multiagente, de acuerdo con el título de los congresos europeos de inteligencia artificial distribuida titulados MAAMAW (*Modelling Autonomous Agents in a Multi-Agent World*, Modelando Agentes Autónomos en un Mundo Multiagente).

1 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, páginas 6

- Inteligencia artificial distribuida “comercial” se centra en la aplicación de la inteligencia artificial distribuida clásica y autónoma, desarrollando agentes (denominados genéricamente agentes software) con características muy diferenciadas (agentes móviles, personales, etc.) que están siendo explotados de forma comercial.

Para presentar las diferentes áreas de trabajo de la inteligencia artificial distribuida, hemos partido de la clasificación realizada por Moulin y Chaib-draaa[2] , que distingue cuatro áreas:

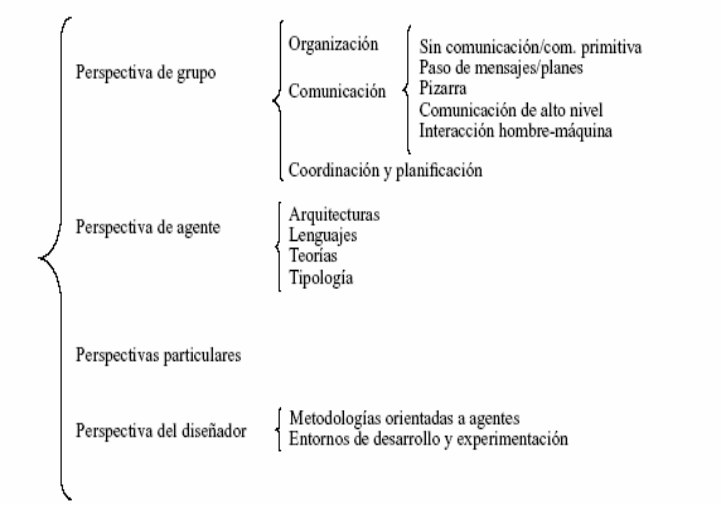
- La perspectiva de grupo, que estudia los elementos que caracterizan a un grupo de agentes, su comunicación e interacciones. Los elementos característicos pueden descomponerse a su vez en tres aspectos: organización del grupo, comunicación y coordinación.
- La perspectiva de agente, que incluye todos los elementos que caracterizan a un agente. La perspectiva de agente ha sido desarrollada siguiendo las divisiones de la “Inteligencia artificial distribuida autónoma y comercial”, combinando la división de los congresos ATAL (Teorías, Arquitecturas y Lenguajes de Agentes, *Agent Theories, Architectures and Languages*) con la tipología de agentes software de Nwana[3].
- Perspectivas específicas, que recogen las relaciones de otros campos con la inteligencia artificial distribuida.
- La perspectiva del diseñador, que recoge los métodos, técnicas de implementación, bancos de prueba, herramientas de diseño y aplicaciones de la inteligencia artificial distribuida.

---

[2] Chaib-draa, B., Moulin, Autores del artículo donde se clasifica la Inteligencia Artificial Distribuida, "Trends in Distributed Artificial Intelligence", www.neoyet.com/Agentes.htm

[3] Hyacinth S. Nwana - Autor de la clasificacion de la tipologia de agentes. Knowledge Engineering Review. Vol. **11**, No 3, pp.1-40, Sept 1996. http://sce.carleton.ca

Figura 1 Marco para analizar y clasificar inteligencia artificial distribuida[4]



1.1.1. Perspectiva de grupo[5]

La inteligencia artificial distribuida "clásica" es un subcampo de la inteligencia artificial dedicada al estudio de las técnicas y el conocimiento necesario para la coordinación y distribución del conocimiento y las acciones en un entorno con múltiples agentes.

Podemos distinguir dos áreas principales de investigación:

- Resolución (cooperativa) de problemas distribuidos ((C) DPS, *(Cooperative) Distributed Problem Solving*) estudia cómo un conjunto de módulos (o "nodos") cooperan para dividir y compartir el conocimiento de un problema y en el desarrollo de la solución.
- Sistemas Multiagente (MAS, *Multiagent Systems*) estudia la coordinación de la conducta inteligente entre un conjunto de (posiblemente pre-existentes) agentes inteligentes autónomos.

4 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 7.

5 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, páginas 8.

La principal diferencia entre ambas áreas estriba en la flexibilidad de la coordinación entre los agentes. En la DPS, las interacciones y tareas que cada agente realiza están prefijadas de antemano: hay un plan centralizado de resolución del problema. Suele haber un miembro que ejerce un control global que centraliza los resultados parciales y datos entre el resto de los componentes del sistema. En contraposición en los MAS, los agentes tienen un grado de autonomía mayor y pueden decidir dinámicamente qué interacciones son adecuadas, qué tareas deben realizar, quién realiza cada tarea y, además, es posible mantener conocimiento que no es globalmente consistente, incluso los agentes pueden mantener objetivos globales diferentes[6].

El término "multiagente" ha alcanzado tal difusión que hoy la inteligencia artificial distribuida aparece como un área de interés dentro de las conferencias de MAS. Sin embargo, esta distinción inicial permitía distinguir entre sistemas que se centraban en el comportamiento global, con una conducta fija de los agentes (DPS) y sistemas que se centraban en la conducta de los individuos que como resultado, obtenían una conducta del sistema (MAS). Si realizamos una analogía con las sociedades humanas, las dos posturas serían resolver un problema con un estado (el diseñador) que planifique y regule las conductas de los individuos (que serán predecibles) o dejar que el sistema se resuelva por la libre iniciativa de los individuos.

Los problemas básicos que estudia inteligencia artificial distribuida clásica y que serán comunes a todos los sistemas, son:

1. Cómo formular, describir, descomponer y asignar problemas y sintetizar los resultados entre un grupo de agentes inteligentes.

6 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 8.

2. Cómo capacitar a los agentes para que se comuniquen e interactúen: qué lenguajes de comunicación o protocolos deben utilizarse, qué y cuándo deben comunicarse, etc.
3. Cómo asegurar que los agentes actúan coherentemente al tomar decisiones o realizar acciones, cómo acomodar los efectos globales de las decisiones locales y prevenir interacciones no deseadas.
4. Cómo capacitar a los agentes para representar y razonar sobre acciones, planes y conocimiento de otros agentes para coordinarse; cómo razonar sobre el estado de su proceso de coordinación (inicio o terminación).
5. Cómo reconocer y reconciliar puntos de vista e intenciones conflictivas entre un conjunto de agentes para coordinar sus acciones; cómo sintetizar los puntos de vista y los resultados.
6. Cómo utilizar técnicas ingenieriles y desarrollar sistemas con inteligencia artificial distribuida. Cómo diseñar plataformas de MAS y metodologías de desarrollo con técnicas de inteligencia artificial distribuida.

Desarrollaremos a continuación los aspectos relacionados con la perspectiva de grupo. La perspectiva de grupo recoge los aspectos de organización entre los agentes, las posibles formas de comunicación y los aspectos de control dinámico para conseguir una conducta global coherente.

1.1.1.1. Descripción, descomposición y asignación de tareas[7]

Uno de los problemas de la inteligencia artificial distribuida es cómo representar los problemas, ya que la descomposición de los mismos depende en gran medida de su formulación. La descripción debe incluir la información sobre las características y atributos del problema así como del dominio y del entorno del problema.

7 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 9.

Una vez dada la descripción de una tarea, la descomposición de una tarea y asignación de subtareas a múltiples agentes debe tener en cuenta que los agentes tengan capacidad para llevarlas a cabo y disponibilidad de recursos[8]. Las dimensiones utilizadas comúnmente para realizar la descomposición de un problema son:

- Nivel de abstracción: los agentes que actúan como resolutores de problemas pueden ser asociados con un nivel de abstracción del problema. La descomposición del problema en niveles de abstracción proporciona una buena base para la descomposición de tareas.
- Dependencias de control: las tareas pueden descomponerse tomando como principio la reducción de las dependencias de datos o control entre tareas, de forma que se restrinja la comunicación. Las dependencias de los datos pueden ser semánticas, temporales o procedentes de la distribución de los datos de entrada (por ejemplo, una red de sensores). Cuando una tarea se distribuye, suele ser necesario introducir tareas de control para coordinarlas.
- División funcional/producto: siguiendo la teoría de la organización, la descomposición funcional consiste en agrupar a los trabajadores que realizan funciones similares, mientras que la división por productos consiste en agrupar a los que trabajan en la producción de un mismo producto.
- Necesidad de redundancia: para garantizar la robustez del sistema o para incluir diferentes perspectivas puede ser necesario duplicar tareas entre diferentes resolutores de problemas. En general, debe evitarse que una tarea sea sólo resoluble por un agente, para evitar los cuellos de botella.

8 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 10.

- Minimización de recursos: debe minimizarse la utilización de los recursos para evitar una sobrecarga de comunicación y coordinación para acceder a dichos recursos limitados.

Las técnicas más empleadas para realizar una descomposición automática de las tareas por parte de los agentes son:

  - Tareas inherentemente descomponibles: la propia descripción de la tarea incluye su descomposición.
  - Descomposición por el programador.
  - Planificación jerárquica: es uno de los sistemas más empleados para descomponer de forma automática. Las tareas se definen en forma de planes que satisfacen unos objetivos y dan lugar a árboles Y-O. Un plan contiene sub-planes.
  - Agregación de subtareas: enfoque ascendente en vez de enfoque descendente en la descomposición.

1.1.1.2. Comunicación[9]

Los agentes pueden mejorar su coordinación y coherencia gestionando qué, cómo y cuándo se comunican entre sí. La comunicación puede proporcionar a los agentes el conocimiento necesario para desarrollar sus acciones con una visión menos local y poder sincronizarse con el resto de agentes. Sin embargo, una excesiva comunicación puede dar lugar a una sociedad de agentes burocráticos, cuya sobrecarga de comunicación sea mayor que el trabajo efectivo realizado. Podemos distinguir un rango amplio de formas de comunicación, que van desde la falta de comunicación hasta la comunicación de alto nivel y la interacción hombre máquina.

9 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 10.

1.1.1.2.1. Sin comunicación[10]

Los agentes pueden interactuar sin comunicarse, infiriendo las intenciones de otros agentes. Esta situación puede darse debido a fallos de hardware, a la imposibilidad de comunicarse o al deseo de que los agentes gocen de una mayor autonomía.

Normalmente, para que sea posible la cooperación se supondrá que los agentes disponen de la información sensorial suficiente para poder inferir los objetivos e intenciones del resto de agentes.

Para estudiar este tipo de interacción se ha recurrido a la teoría de juegos, empleado matrices de costes, en las que se representa la ganancia que obtiene cada agente dependiendo de la acción que realice otro agente. Este enfoque se ha extendido con técnicas probabilísticas para representar la incertidumbre sobre los posibles movimientos de los otros agentes.

1.1.1.2.2. Comunicación primitiva[11]

La comunicación se restringe a un número de señales con interpretaciones fijas (por ejemplo el uso de las señales enviar/recibir en CSP (*Communicating Sequential Processes*). Este trabajo ha sido aplicado a planificación multiagente para coordinar a dos agentes que tienen un conflicto (de escasez de recursos) mediante un mediador, pero la coordinación que puede lograrse es limitada (ceder el turno a través de un mediador que sincroniza a los agentes).

---

10 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 10.

11 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 11.

1.1.1.2.3 Arquitectura de pizarra[12]

La arquitectura de pizarra[13] es otro paradigma de comunicación, cuya estructura consta tres componentes principales: la pizarra, un conjunto de fuentes de conocimiento (KSs, *Knowledge sources*) y un mecanismo de control. La pizarra es una base de datos global (compartida por todas las KSs) que contiene datos e hipótesis (soluciones parciales potenciales). Se suele estructurar la pizarra en niveles, asociando clases de hipótesis a un nivel, y las hipótesis suelen estar ligadas a hipótesis de otro nivel.

Las fuentes de conocimiento son módulos especializados que intercambian mensajes (hipótesis, resultados parciales, etc.) a través de la pizarra. Partiendo de los resultados e hipótesis de otras KSs, una KS puede generar otra solución parcial tentativa. Cada nivel de la pizarra suele suponer un nivel de abstracción en la resolución del problema, y las KSs trabajando en un nivel ven las hipótesis de su nivel y los adyacentes, de forma que las hipótesis de los niveles próximos se van propagando a los niveles más abstractos de forma ascendente. Para gestionar la concurrencia en el acceso a la pizarra se debe emplear un mecanismo de control, típicamente basado en una agenda.

Esta agenda es gestionada por un monitor de la pizarra que activa las fuentes de conocimiento adecuadas según los datos mantenidos en la pizarra. Algunas arquitecturas de agente utilizan esta arquitectura para intercambiar información entre subsistemas de agentes o para modelar el intercambio entre los distintos módulos (o subagentes) que componen la estructura de un agente.

---

12 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 11.

13 El termino pizarra proviene de una metáfora, que compara esta estructura con un grupo de personas que resuelve un problema mediante la tecnica “tormenta de ideas”.

1.1.1.2.4. Paso de mensajes[14]

Los sistemas de paso de mensajes permiten que un agente envíe un mensaje (tanto de peticiones de servicios e información como de respuesta a dichas peticiones) a uno o más agentes cuyos nombres debe conocer. A diferencia de la arquitectura de pizarra, los agentes deben mantener conocimiento sobre su entorno para saber a qué agentes deben dirigir sus mensajes. Este modelo tiene su origen en la programación orientada a objetos concurrente.

Numerosos sistemas multiagente han adoptado el paso de mensajes definiendo un formato para dichos mensajes, tipos de mensaje y un protocolo para procesar dichos mensajes,

1.1.1.2.5. Comunicación de alto nivel[15]

Se ha realizado un gran esfuerzo investigador para estudiar las interacciones entre los agentes en el nivel de conocimiento en vez de en el nivel simbólico. Esto supone que los agentes puedan razonar sobre las intenciones, deseos y objetivos de otros agentes, y que puedan comunicar estos deseos, objetivos e intenciones. Para analizar estas interacciones complejas, se han intentado aplicar a los sistemas multiagente algunas técnicas y teorías provenientes del campo del lenguaje natural, en especial de análisis y generación del discurso. De acuerdo con las técnicas simbólicas de comprensión del diálogo, no basta con analizar el significado de cada frase para entender un texto, sino que es necesario comprender las intenciones de los interlocutores y cómo se desarrolla el diálogo para satisfacer sus objetivos, ya que no toda la comunicación suele ser explícita.

---

14 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 11.

15 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 12.

Como marco teórico, se ha adoptado frecuente la teoría de los actos de habla (*speech-acts*) que ha sido aplicada notablemente en el lenguaje de comunicación para agentes KQML (Lenguaje de consulta y manipulación del conocimiento, *Knowledge Query and Manipulation Language*).

1.1.1.2.6. Interacción hombre-máquina[16]

La comunicación entre un agente artificial y un agente humano ha tenido en los últimos tiempos gran relevancia. Básicamente, se han tomado dos aproximaciones: encapsular al agente humano modelando sus interacciones en un lenguaje de comunicación de agentes o aprovechar la tecnología multiagente para simplificar las interfaces hombre-máquina.

1.1.1.3. Coherencia y coordinación[17]

Definimos coherencia como la propiedad de un sistema para comportarse como una unidad, de acuerdo con alguna dimensión de evaluación. Podemos evaluar la coherencia de un sistema examinando varias dimensiones de su conducta:

- Calidad de la solución: habilidad del sistema para alcanzar soluciones satisfactorias, y la calidad de las soluciones que produce. Requiere que se alcancen tres condiciones:
    - Cobertura: cada tarea necesaria del problema debe ser realizada al menos por un nodo.
    - Conectividad: los nodos deben interactuar de forma que las actividades cubiertas puedan ser desarrolladas e integradas en una solución global.

16 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 12.

17 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 12.

    - Capacidad: la cobertura y la conectividad deben estar disponibles en la comunicación.
- Eficiencia: eficiencia global del sistema para alcanzar un fin.
- Claridad: claridad conceptual de las acciones del sistema y utilidad de su representación. Posibilidad de describir y representar la conducta del sistema de forma que un observador externo pueda entenderla. En un sistema describible y bien estructurado la auto-representación puede ser usaba para comunicación interna, reorganización, diagnóstico de fallos, análisis de rendimiento, etc.
- Robustez: grado de degradación del sistema en presencia de fallos o incertidumbre.

Definimos coordinación como la propiedad de interacción entre un conjunto de agentes que realizan alguna actividad colectiva[18]. El grado de coordinación exhibido por un conjunto de agentes es el área en que pueden evitar realizar un trabajo de articulación para coordinarse. La coordinación efectiva implica un cierto grado de predecibilidad mutua y falta de conflictos. Cuantos más conflictos inesperados se den, peor coordinados estarán los agentes. Definimos cooperación como una clase de coordinación entre agentes no antagonistas (los agentes antagónicos pueden ser coordinados, por ejemplo, si son robots para no chocar). La coherencia y la coordinación están relacionadas: una mejor coordinación debe guiar a una mayor coherencia de eficiencia, por la reducción del trabajo de articulación.

Podemos distinguir los siguientes mecanismos para facilitar la coordinación[19]:

18 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 13.

19 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 13.

- Negociación: empleo de diálogo entre los nodos para resolver vistas inconsistentes y alcanzar un acuerdo sobre cómo trabajar conjuntamente.
- Cooperación funcionalmente precisa (FA/C, *Functionally accurate cooperative*): la inconsistencia se supera intercambiando soluciones tentativas para resolver errores y converger en las soluciones del problema.
- Estructuración organizativa: utilización de conocimiento común sobre los papeles generales de resolución del problema y patrones de comunicación para reducir la incertidumbre de los nodos y sobre cómo deben cooperar.
- Planificación multiagente: compartición de información para construir un plan de cómo los agentes deben trabajar juntos, distribuyendo y siguiendo este plan durante la resolución del problema.
- Entornos teóricos: utilización de modelos lógicos y matemáticos de los agentes, de sus creencias y de su razonamiento para comprender las capacidades teóricas de las redes de agentes.
- Control local sofisticado: integración del razonamiento sobre las acciones y creencias de otros agentes con el razonamiento local sobre la resolución del problema, de forma que las decisiones de coordinación sean parte de las decisiones locales en vez de una capa separada sobre la resolución local del problema.

Salvo en el caso de planificación, no se discuten casos de coordinación centralizada, ya que esta solución no es en general viable, debido a su intratabilidad computacional (la carga de un solo coordinador coordinando a muchos nodos es muy alta), a la comunicación limitada (el coordinador constituiría un cuello de botella) y a la falta de robustez (el rendimiento de la red no debe depender de un solo nodo).

1.1.1.4. Negociación[20]

La negociación puede definirse como el proceso de mejorar el acuerdo (reduciendo inconsistencias e incertidumbre) sobre puntos de vista comunes a través del intercambio estructurado de información relevante.

- Lenguaje: trata de las primitivas de comunicación para negociar (proponer, refinar, confirmar, etc.), su semántica, el objeto de la negociación (plan, oferta de tarea, etc.) y los protocolos de negociación.
- Decisión: trata de qué aspectos tienen en cuenta los agentes para decidir en la negociación como maximizar una función de utilidad, preferencias, estrategias de negociación (ser cooperativos, competitivos, etc.).
- Proceso: estudio de modelos generales del proceso de negociación y de la conducta global de los participantes.

A continuación analizaremos algunos de los enfoques específicos de negociación más conocidos. La negociación ha sido empleada para asignación de tareas, asignación de recursos, resolución y prevención de conflictos. Se ha empleado negociación bajo diferentes supuestos: agentes no-cooperativos, agentes cooperativos, a través de un mediador o jerárquicamente en una organización.

1.1.1.4.1. Negociación para asignación de tareas[21]

El protocolo de contratos (CNP, *Contract Net Protocol*) es uno de los protocolos más conocidos en los sistemas multiagente. Es un protocolo de alto nivel que facilita el control distribuido de la ejecución de tareas de forma cooperativa.

---

20 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 14.

21 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 14.

El principal problema que resuelve es el problema de conexión entre nodos que deben ejecutar tareas y nodos que están inactivos. Este problema se resuelve utilizando un mecanismo de mercado: realizar contratos entre nodos. El nodo que desea contratar a otros nodos actúa como gestor, y los nodos que acepten realizar una tarea actuarán como contratados. Un gestor es responsable de supervisar la ejecución de una tarea y procesar los resultados de la ejecución. Un contratado es responsable de realizar una tarea. Los nodos no se determinan a priori, sino dinámicamente, y un nodo puede tener ambos papeles (gestor/contratado) en diferentes contratos.

El proceso de negociación consiste en que el gestor envía anuncios de tareas a los posibles contratados disponibles. Estas ofertas son evaluadas por los posibles contratados que envían ofertas (conteniendo, por ejemplo, el coste) para las tareas que pueden realizar. Dichas ofertas son evaluadas por el gestor que acepta (firmando un contrato) o rechaza. Un nodo que ha recibido una tarea puede subcontratar la tarea o parte de ella, adoptando el papel de gestor y anunciando dicha sub-tarea. El protocolo normaliza otras interacciones para paso de informes internos o finales (la solución) y diversos casos particulares.

1.1.1.4.2. Negociación para asignación de recursos[22]

La negociación multiestado extiende la negociación de tareas de CNP para los problemas en que la escasez de recursos pueda determinar que no todos los objetivos globales puedan cumplirse.Extiende el protocolo CNP básico para permitir una negociación iterativa en la oferta y asignación de tareas.

22 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 14.

Cada nodo indica a qué tareas puede comprometerse y a cuáles no, de forma que los nodos converjan en soluciones globalmente posibles, pero sin tener una visión de la solución global. El ejemplo de aplicación es una red de comunicaciones para restauración de circuitos en la que los nodos de control tienen limitaciones para asignar más de dos circuitos en cada enlace y no todas las posibles caídas de enlaces pueden restaurarse.

1.1.1.4.3. Negociación para resolución de conflictos[23]

En el entorno de control aéreo, se proponen el uso de negociación para determinar quién resuelve un conflicto. Cuando se detecta un conflicto entre agentes (p.ej. los aviones controlados están muy cerca), los agentes emplean la centralización de tareas, que consiste en elegir a un agente para que resuelva el conflicto. Este agente elegido (con diferentes criterios: el que tiene mayor conocimiento, menor carga o por convenio previo) tiene que replanificar el sistema, retransmitir este plan y ejecutarlo.

1.1.1.4.4. Modelado cognitivo de la negociación[24]

Modelo computacional para resolver conflictos vía negociación, destinado a alcanzar un compromiso entre múltiples agentes en uno o más encuentros sobre uno o más temas.

Permite generar una propuesta de un compromiso, argumentos persuasivos, razones en contra y a favor del acuerdo, peticiones de informaciones adicionales y medidas de la utilidad de los agentes para acordar o no. Se emplea razonamiento basado en casos para aprender de experiencias pasadas.

---

23 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 15.

24 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 15.

En un mocelo de negociación cuando hay limitacion de recursos se consideran tres operadores: composición (crear una nueva alternativa combinando dos alternativas previas; suele ser debido a dependencias condicionales entre las alternativas previas; p.ej. necesidad de dos recursos, el segundo requiere el primero y se tratan como un bloque), reconfiguración (modificar una alternativa para satisfacer a ambos negociadores; normalmente un agente quería ofrecer A, y ahora debe ofrecer A y B para satisfacer al otro agente) y relajación (no se pueden conseguir los requisitos y hay que relajarlos progresivamente en un proceso de regateo). Para modelar la negociación como un proceso de búsqueda distribuida de una solución entre varios agentes: cada agente va refinando una solución inicial y relajando las restricciones débiles, siguiendo la filosofía de las arquitecturas de pizarra.

1.1.1.5. Cooperación funcionalmente precisa[25]

El paradigma FA/C (Sistemas Distribuidos Cooperativos, funcionalmente precisos, *Functionally Accurate, Cooperative Distributed Systems*) se distingue porque los nodos de la red mantienen incertidumbre acerca de sus visiones del problema. Para avanzar en la resolución del problema, cada nodo debe cooperar con el resto para detectar las inconsistencias entre sus resultados parciales tentativos y los recibidos de otros nodos, e integrar estos resultados con los suyos. Los sistemas convencionales seguirían el paradigma CA/NA (Sistemas distribuidos completamente precisos, semi-autónomos; *Completely accurate, nearly autonomous*), porque cada nodo mantiene una información precisa y en caso de no tenerla, la demanda a otro nodo. Los sistemas más conocidos de este enfoque son los sistemas de pizarra Hearsay-II (Entorno de supervisión de vehículos distribuidos, *Distributed Vehicle Monitoring Testbed*).

25 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 16.

1.1.1.6. Estructuración organizativa[26]

Una diferencia importante entre la negociación y la cooperación funcionalmente precisa es que la negociación toma un enfoque descendente (*top-down*) mientras que la cooperación funcionalmente precisa toma un enfoque ascendente (*bottom-up*). La estructuración organizativa se sitúa entre ambas posiciones. Una estructura organizativa es un patrón de las relaciones de control e información entre los nodos, y la distribución de las capacidades de resolución del problema entre los mismos. Mientras que la negociación crea alianzas temporales entre los nodos, la estructura organizativa representa unas relaciones más permanentes a largo plazo.

La estructura organizativa puede mejorar la coordinación en un sistema cooperativo, funcionalmente preciso, dotando a cada nodo de una visión de alto nivel de cómo la red resuelve los problemas y el papel que cada nodo juega en dicha red.

Podemos distinguir los siguientes tipos organzativos.

- Organización jerárquica o centralizada: la autoridad para la toma de decisiones y el control se concentra en un resolutor o en un grupo especializado, en cada nivel de la jerarquía.
- Comunidad con reglas de conducta: se definen unas reglas o pautas de conducta para la relación entre los nodos, sin una organización explícita.
- Organización de mercado: el control se distribuye como en un mercado, y los nodos compiten por las tareas y recursos realizando ofertas y contratos, o evaluación económica de los servicios y la demanda.
- Comunidad plural: se toma la comunidad científica como modelo, y las soluciones locales se publican y se van refinando por la comunidad.

26 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 16.

1.1.1.7. Planificación multiagente[27]

La planificación multiagente trata de facilitar la cooperación entre los nodos mediante la creación de un plan que especifica las acciones e interacciones futuras de los agentes.

Se diferencia de la negociación en que el plan multiagente trata de garantizar la consistencia global, mientras que las negociaciones bilaterales pueden llevar a situaciones de conflicto. Dependiendo de si el plan es generado por un agente o entre todos los agentes, se distingue entre planificación centralizada y distribuida.

- Planificación centralizada: se basa en la presencia de un nodo que actúa como coordinador, definiendo un plan global a partir de la información y/o planes individuales de los nodos.
- Planificación distribuida: el principio general es que cada agente planifique sus acciones teniendo una visión (en general, parcial) de los planes del resto de agentes. Durante esta planificación, los agentes pueden negociar para alcanzar compromisos. Destaca el algoritmo de planificación parcial global (PGP, *Partial Global Planning*), que se basa en que cada nodo construya un plan local, lo comparta con otros agentes, e intente identificar y construir planes globales parciales, que capturen objetivos globales a varios agentes. Una vez construidos los planes globales parciales, se planifican las acciones locales y las comunicaciones con el resto de agentes involucrados en el plan.

  Después se ejecuta el plan, observando el entorno para ver si se desvía de lo planificado y es necesario volver a planificar.

---

27 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 16.

1.1.1.8. Reconocimiento y resolución de discrepancias entre agentes[28]

Se pueden distinguirse cuatro tipos de disparidades de conocimiento: incompletitud, sucede cuando un agente tiene algún conocimiento que otro no tiene; inconsistencia, sucede cuando dos agentes tiene diferentes valores de verdad para la misma proposición lógica; incompatibilidad, sucede cuando el conocimiento es representado en formas incompatibles (por ejemplo, marcos con diferentes ranuras y semánticas); e inconmensurabilidad, que sucede cuando el conocimiento está representado de la misma forma, pero las interpretaciones semánticas son diferentes.

Para reconocer las disparidades es necesario que se pueda representar la compatibilidad. Suelen representarse las creencias de los agentes. Algunos de los métodos empleados para reconciliar las discrepancias o conflictos son:

- Obtención de un conocimiento común: cuando la disparidad es debida a conocimiento incompleto, puede solicitarse este conocimiento a otros agentes.
- Revisión de premisas: cuando se detectan proposiciones inconsistentes, pueden revisarse las premisas de dichas proposiciones para descubrir si son estas premisas las causas de la inconsistencia.
- Autoridad y mediación: en numerosos casos, es necesario el arbitrio de un mediador o un criterio jerárquico para resolver el conflicto.
- Resolución basada en casos: los conflictos pueden resolverse recurriendo a casos similares sucedidos en el pasado.

28 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 17.

- Resolución de restricciones: en el caso de que el conflicto se dé por restricciones conflictivas, el conflicto puede resolverse relajando las restricciones no esenciales.
- Negociación: la negociación suele ser una técnica empleada para resolver conflictos.
- Estandarización: la experiencia acumulada en la resolución de conflictos puede conducir a la estandarización de las conductas para evitar o resolver los conflictos.

1.1.2. Perspectiva del agente[29]

La aquí denominada "Inteligencia Artificial Distribuida autónoma" se centra en los micro-aspectos de la inteligencia artificial distribuida, es decir, en los agentes inteligentes, más que en los macro-aspectos (tratados en la inteligencia artificial distribuida clásica), si bien estos macro-aspectos son relevantes para abordar los micro-aspectos. Dichos aspectos se dividen en tres áreas: teoría de agentes, que trata de responder a la pregunta de qué es un agente y de la utilización de formalismos matemáticos para representar y razonar sobre las propiedades de agentes; arquitecturas de agentes, que trata de las arquitecturas software/hardware que permiten reflejar las propiedades enunciadas por los teóricos; y lenguajes de agentes, que son los sistemas software para programar y experimentar con agentes.

Como complemento a estos aspectos, añadimos un cuarto enfoque: la tipología de agentes, que añade algunos agentes cuya característica es su campo de aplicación, más que su arquitectura.

29 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 18.

1.1.2.1. Teoría de agentes[30]

Las teorías de agentes son especificaciones para conceptualizar los agentes. Debido a que la definición de agente ha resultado ser tan controvertida como la definición de inteligencia artificial, se ha optado por una definición de un conjunto de propiedades que caracterizan a los agentes, aunque un agente no tiene que poseer todas estas propiedades:

- Autonomía: los agentes pueden operar sin la intervención de humanos o de otros agentes;
- Sociabilidad: los agentes son capaces de interaccionar con otros agentes (humanos o no) a través de un lenguaje de comunicación entre agentes;
- Reactividad: los agentes son capaces de percibir estímulos de su entorno y reaccionar a dichos estímulos;
- Proactividad, iniciativa: los agentes no son sólo entidades que reaccionan a un estímulo, sino que tienen un carácter emprendedor, y pueden actuar guiados por sus objetivos;
- Movilidad: capacidad de un agente de trasladarse a través de una red telemática;
- veracidad: asunción de que un agente no comunica información falsa a propósito;
- Benevolencia: asunción de que un agente está dispuesto a ayudar a otros agentes si esto no entra en conflicto con sus propios objetivos; y
- Racionalidad: asunción de que un agente actúa de forma racional, intentando cumplir sus objetivos si son viables.

La cuestión de qué es un agente, como hemos comentado, está aún siendo debatida, corriendo el riesgo de que cualquier programa sea denominado agente.

30 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 18.

Se pueden distinguir dos nociones extremas de agentes:

- Una noción débil de agente consiste en definir un agente como a una entidad que es capaz de intercambiar mensajes utilizando un lenguaje de comunicación de agentes. Esta definición es la más utilizada dentro de la ingeniería software basada en agentes, cuyo fin es conseguir la interoperabilidad entre aplicaciones a nivel semántico utilizando la emergente tecnología de agentes.
- Una noción más fuerte o restrictiva de agente es la enunciada por Shoham en su propuesta de programación orientada a agentes (AOP), donde un agente se define como una entidad cuyo estado es visto como un conjunto de componentes mentales, tales como creencias, capacidades, elecciones y acuerdos.

Los agentes suelen ser considerados como sistemas intencionales, esto es, sistemas cuya conducta puede ser predecida atribuyendo creencias, deseos y una conducta racional. Para representar estas intenciones, se han empleado diversos formalismos lógicos, de entre los que cabe destacar la teoría de la intención de Cohen y Levesque, la lógica multi-modal BDI (Creencia, Deseo e Intención; *Belief, Desire, Intention*) y la familia de lógicas para especificar sistemas multiagente propuestas por Wooldridge[31].

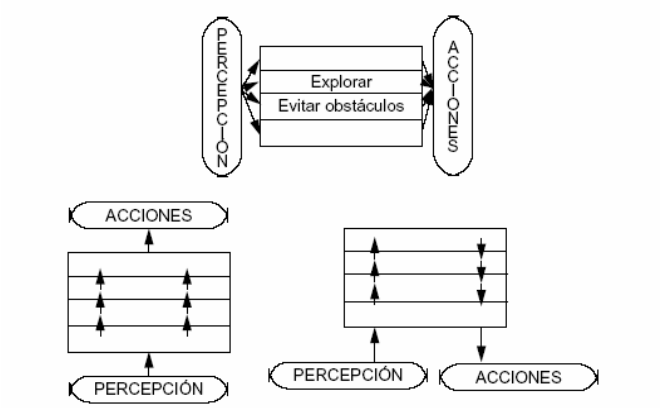
El modelo formal BDI de Rao y Georgeff incluye la definición de la lógica proposicional, temporal y multimodal, descripción de operadores de creencias (BEL), deseos (DES) e intenciones (INTEND), la definición de una semántica de mundos posibles para estos operadores y la definición axiomática de la interrelación y propiedades de estos operadores BDI.

31 D. Carlos Carrascosa Casamayor, "Meta-razonamiento en Agentes con Restricciones Temporales Críticas", http://www.dsic.upv.es/docs/bib-dig/tesis/etd-12162003-124543/tesisCarlosCarrascosa.pdf, páginas 14-23.

1.1.2.2. Arquitecturas de agentes[32]

Las arquitecturas de agentes describen la interconexión de los módulos software/hardware que permiten a un agente exhibir la conducta enunciada en las teorías de agentes[33]. Frente a otras tecnologías con componentes fijos como la de objetos (atributos y métodos) o la de sistemas expertos (motor de inferencias, base de conocimiento y otros elementos opcionales), en los agentes nos encontramos con una gran variedad de arquitecturas. Una primera clasificación de las arquitecturas puede ser realizada según todas las capas tengan acceso a sensores y actuadores (horizontales) o sólo la capa más baja tenga acceso a sensores y actuadores (verticales), tal como se muestra en la figura 2. Las arquitecturas verticales ofrecerán la ventaja del paralelismo entre capas a costa de un alto conocimiento de control para coordinar las capas, mientras que las verticales reducen este control a costa de una mayor complejidad en la capa que interactúa con los sensores. También podemos clasificar las arquitecturas atendiendo al tipo de procesamiento empleado en deliberativas, reactivas e híbridas.

Figura 2 Arquitecturas horizontales y verticales



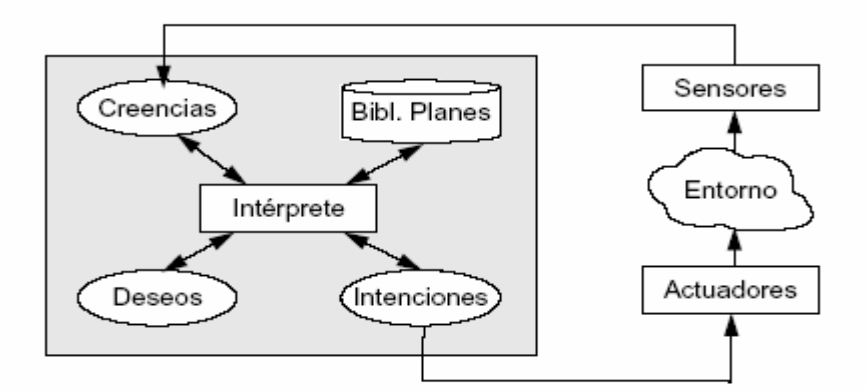
32 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 20.

33 Muñoz Salinas, Rafael, "Soft-Computing and Computer Vision Techniques Aplied to Autonomos Robot Navigation and Human-Robot Interaction", http://hera.ugr.es/tesisugr/16094347.pdf, páginas 37-42.

1.1.2.1.2. Arquitecturas deliberativas[34]

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

Figura 3 Arquitectura de un agente deliberativo



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

34 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 21.

Podemos distinguir los siguientes tipos principales de arquitecturas deliberativas o simbólicas, arquitecturas intencionales y arquitecturas sociales. Las agentes intencionales se distinguen por ser capaces de razonar sobre sus creencias e intenciones. Se pueden considerar como sistemas de planificación que incluyen creencias e intenciones en sus planes. Los agentes sociales se pueden definir como agentes intencionales que mantienen además un modelo explícito de otros agentes y son capaces de razonar sobre estos modelos.

Dentro de las arquitecturas intencionales, cabe destacar aquellas que han tomado como punto de partida la teoría de agentes BDI en su implementación, representando explícitamente las actitudes intencionales de los agentes. Estos sistemas también suelen utilizar planificación para determinar qué acciones deben llevar a cabo pero, a diferencia de los agentes planificadores, emplean planes en que se comprueban creencias, deseos e intenciones.

Las creencias son el conocimiento que el agente tiene sobre sí mismo y su entorno. Los deseos son objetivos del agente a largo plazo que desea cumplir. Como normalmente no puede cumplir todos los objetivos a la vez, ya que tiene unos recursos limitados, se introducen las intenciones, que son los objetivos que en cada momento intenta cumplir el agente. Normalmente también se introduce el concepto de planes, que permiten definir las intenciones como los planes que un agente está realizando en un momento dado. Hay muchas arquitecturas de agentes que siguen el modelo BDI, como IRMA y PRS. Los agentes sociales pueden clasificarse en dos grandes grupos: agentes intencionales cuya arquitectura ha sido aumentada para abordar el razonamiento sobre otros agentes, como COSY, GRATE y DA-Soc;

Y arquitecturas que siguiendo la inteligencia artificial distribuida clásica han prestado más atención a los aspectos cooperativos (cuándo, cómo y con quién cooperar), sin modelar necesariamente las intenciones de los agentes, como Archon, Imagine, Coopera y MAST.

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

1.1.2.1.3. Arquitecturas reactivas[35]

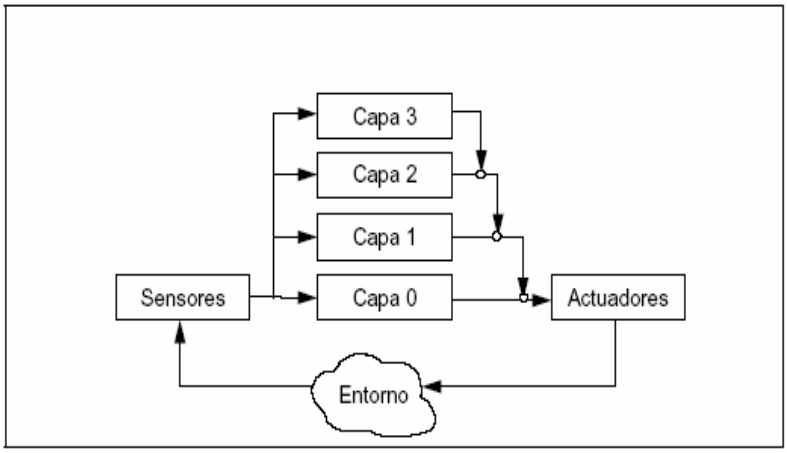
Las arquitecturas reactivas cuestionan la viabilidad del paradigma simbólico y proponen una arquitectura que actúa siguiendo un enfoque conductista, con un modelo estímulo-respuesta. Las arquitecturas reactivas no tienen un modelo del mundo simbólico como elemento central de razonamiento y no utilizan razonamiento simbólico complejo, sino que siguen un procesamiento ascendente (bottomup), para lo cual mantienen una serie de patrones que se activan bajo ciertas condiciones de los sensores y tienen un efecto directo en los actuadores. Esta discusión entre mantener una representación explícita del modelo o no, no es una discusión específica del campo de agente sino de la inteligencia artificial en general, de hecho las primeras arquitecturas de agentes reactivos se basan en los planificadores reactivos.

35 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 22.

Las principales arquitecturas reactivas son:

- Reglas situadas: la implementación más sencilla de reactividad consiste en definir el comportamiento con reglas del tipo si situación-percibida entonces acciones específicas.

Figura 4 Arquitectura de subsunción de un agente reactivo



- Arquitecturas de subsunción (subsumption) y autómatas de estado finito: permiten gestionar problemas de mayor complejidad que las reglas. Las arquitecturas de subsunción están compuestas por capas que ejecutan una determinada conducta (p.ej. explorar, evitar un obstáculo, etc.). La estructura de cada capa es la de una red de topología fija de máquinas de estado finitas. Las capas mantienen una relación de inhibición sobre las capas inferiores (inhibir entradas de los sensores y acciones en los actuadores). El control no es central, sino dirigido por los datos en cada capa.

- Tareas competitivas: un agente debe decidir qué tarea debe realizar de entre varias posibles, seleccionando la que proporciona un nivel de activación mayor. Se basa en una aproximación ecológica del problema de resolución distribuida de problemas, simulando, por ejemplo en el sistema MANTA, que cada agente es una hormiga y decide qué acción debe hacer para cumplir sus objetivos. El problema se resuelve sin comunicación entre los individuos, estableciendo un criterio de terminación del problema. Por ejemplo, los problemas clásicos de búsqueda (misioneros y caníbales, mundo de los bloques, etc.) se interpretan como agentes (cada misionero, cada bloque, etc.) que pueden realizar movimientos y una condición global de terminación.
- Redes neuronales: la capacidad de aprendizaje de las redes neuronales también ha sido propuesta en algunas arquitecturas formadas por redes capaces de realizar una función (evitar colisiones, etc.).

Las arquitecturas reactivas pueden clasificarse como verticales porque los estímulos recibidos del exterior son procesados por capas especializadas que directamente responden con acciones a dichos estímulos y pueden inhibir las capas inferiores.

1.1.2.1.4 Arquitecturas híbridas[36]

Estas arquitecturas pretenden combinar aspectos deliberativos y reactivos. Los más claros exponentes de estas arquitecturas son PRS, TouringMachines e Interrap. Estas arquitecturas combinan módulos reactivos con módulos deliberativos. Los módulos reactivos se encargan de procesar los estímulos que no necesitan deliberación, mientras que los módulos deliberativos determinan qué acciones deben realizarse para satisfacer los objetivos locales y cooperativos de los agentes.

36 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 24.

Las arquitecturas híbridas pueden ser horizontales (TouringMachines) y verticales (PRS e Interrap)

Figura 5 Arquitectura híbrida (vertical) de Interrap

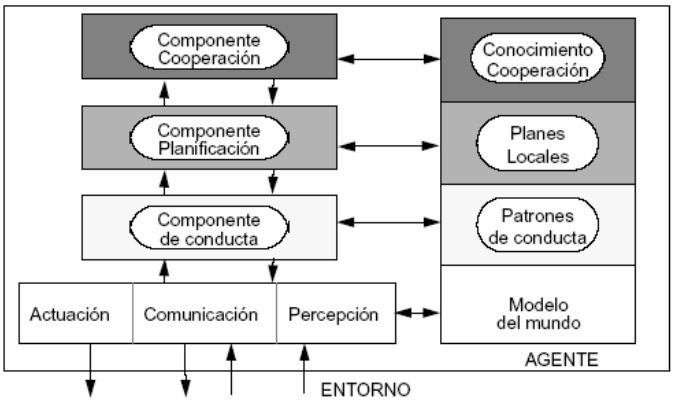
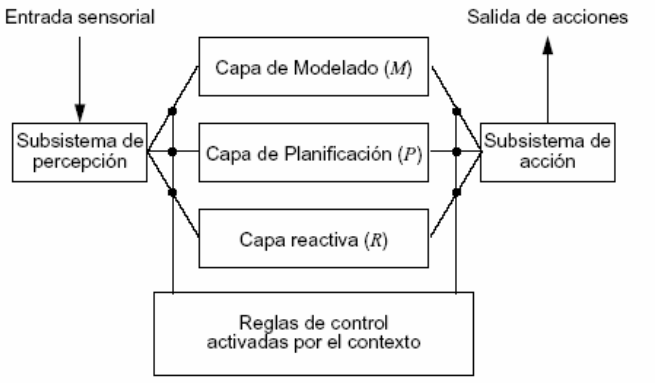


Figura 6 Arquitectura híbrida (horizontal) TouringMachine



1.1.2.2. Lenguajes de agentes[37]

Los lenguajes de agentes se definen como lenguajes que permiten programar agentes con los términos desarrollados por los teóricos de agentes. Podemos distinguir dos tipos principales de lenguajes de programación:

- Lenguajes de agentes de propósito general: lenguajes destinados a programar agentes genéricos utilizables en cualquier aplicación.

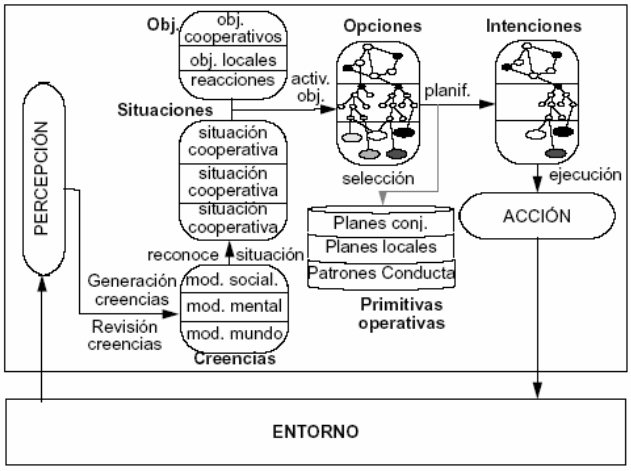
37 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 24.

- Lenguajes de agentes específicos: lenguajes para un tipo de agentes específicos, por ejemplo los lenguajes para agentes móviles Telescript o Agent-Tcl.

Se pueden distinguir los siguientes niveles en la programación de agentes:

- Lenguajes de programación de la estructura del agente: permiten programar las funcionalidades básicas para definir a un agente: funciones de creación de procesos (creación del proceso agente y de los procesos concurrentes con él) y funciones de comunicación entre agentes (nivel de transporte).
- Lenguajes de comunicación de agentes: definición del formato de los mensajes intercambiados, de las primitivas de comunicación y de los protocolos disponibles.
- Lenguajes de programación del comportamiento del agente: permiten definir el conocimiento del agente: conocimiento inicial (modelo de entorno, creencias, deseos, objetivos), funciones de mantenimiento de dicho conocimiento (reglas, planes, etc.), funciones para alcanzar sus objetivos (planes, reglas, etc.) y funciones para desarrollar habilidades (programación de servicios).

Figura 7 Arquitectura conceptual de Interrap



1.1.2.2.1. Lenguajes de programación de la estructura del agente[38]

Este nivel de programación normalmente sólo es utilizado por los desarrolladores de una plataforma de desarrollo de agentes. Los lenguajes empleados suelen ser lenguajes de propósito general (C, C++, Java, Lisp, Prolog, etc.) o lenguajes específicos (p.ej. April dentro del proyecto IMAGINE, sobre Prolog/C y CUBL (Concurrent Unit Based Language) dentro del proyecto DAISY, sobre CLOS/C++).

1.1.2.2.2. Lenguajes de comunicación de agentes[39]

Podemos distinguir dos tipos de lenguajes de comunicación:

- Procedimentales: se basan en el intercambio de directivas procedimentales, es decir, un agente recibe un mensaje que implica la ejecución de un procedimiento. Suelen emplear lenguajes de intérpretes de órdenes (scripts) como Perl, Tcl, etc., permitiendo un rápido prototipazo aunque no suelen ser fácilmente escalables ni reciclables. Son especialmente útiles para la construcción de agentes en aplicaciones finales como agentes de usuario o agentes móviles. Dentro de este enfoque podríamos encontrar a Sodabot o Telescript.
- Declarativos: se basan en el intercambio de actos comunicativos, es decir, un agente recibe un mensaje con un acto comunicativo que le permite interpretar el contenido del mensaje. El ejemplo más extendido de este enfoque es KQML.

38 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 26.

39 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 26.

1.1.2.2.3. Lenguajes de programación del comportamiento del agente[40]

Los lenguajes de descripción de agentes permiten la programación de los agentes: definición de su estructura, conocimiento y habilidades.Podemos distinguir diferentes tipos de lenguajes de descripción de agentes:

- Lenguajes de descripción de agente: los agentes se derivan de una clase de agente genérica, permitiendo la definición de los elementos básicos del modelo de agente tales como base de conocimiento, grupos de agentes, habilidades del agente, servicios ofrecidos, planes para alcanzar objetivos, etc. La descripción se traduce a un lenguaje ya ejecutable. Podemos citar a MACE ADL, AgentSpeak y MAST/ADL.
- Lenguajes de programación orientados a agentes: siguiendo el paradigma de Shoham de la programación orientada a agentes (POA), se han definido algunos lenguajes que tienen en cuenta el estado mental del agente para programar las funciones de transición entre estos estados mentales, que consisten en creencias, capacidades y obligaciones. Entre estos lenguajes podemos citar AGENT0, PLACA y Agent-K.
- Lenguajes basados en reglas de producción: la programación de la base de conocimiento de los agentes se realiza en algunos sistemas empleando reglas de producción como, por ejemplo, MAGSY (basado en OPS5), DYNACLIPS (arquitectura de pizarra basada en CLIPS) y RTA, que permite definir las conductas del agente con reglas.
- Lenguajes de especificación: emplean una especificación (normalmente lógica) del agente que se ejecuta directamente para generar su conducta, pudiendo verificar propiedades de la especificación. Podemos citar como ejemplo a METATEM y DESIRE.

---

40 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 27.

1.1.2.2.4. Tipología de agentes[41]

La metáfora de los agentes ha superado recientemente el dominio de los laboratorios, interesando a varias empresas en el desarrollo de productos comerciales basados en agentes. La principal diferencia entre estos agentes (denominados agentes software) radica en su especialización: están especialmente diseñados para realizar un tipo de tarea.

Son agentes de acuerdo con la noción débil de agente, que se caracterizan por su proactividad, autonomía, aprendizaje o cooperativismo. En este apartado vamos a revisar brevemente algunos de los tipos de agentes más conocidos:

- Agentes de interfaz: los agentes de interfaz, también denominados asistentes personales o agentes de usuario, tienen como objetivo simplificar las tareas rutinarias que realiza un usuario. Pueden, por ejemplo, aprender a filtrar el correo electrónico fijándose en el comportamiento habitual del usuario; planificar encuentros, negociando con los asistentes personales de los otros miembros del encuentro; o detectar que una noticia puede ser relevante para un usuario y comunicárselo.
- Agentes móviles: representan un nuevo paradigma en la computación distribuida. El concepto de movilidad significa que un agente puede transitar entre varias máquinas, para evitar una sobrecarga de comunicación o utilizar recursos de los que no dispone en su máquina. En el lenguaje de programación de agente, tendremos órdenes como “encontrarse” (meet) con otros agentes, y se puede programar cuándo interesar viajar a otra máquina o no. Los principales problemas que surgen son de seguridad.

41 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 27.

- Agentes de internet/información: la gran cantidad de información disponible en Internet ha favorecido el desarrollo de los agentes de información, cuya misión es navegar por la red recolectando información, indexarla y ofrecer la información que puede interesar al usuario cuando realiza una consulta. Un tipo de agentes de Internet son los softbots que ofrecen una interfaz amigable de Unix, permitiendo el cumplimiento de objetivos de alto nivel (p.ej. obtén los artículos de este año de Etzioni; el softbot tendría que buscar dónde están estos artículos, o solicitarlos por correo electrónico, etc.).

1.1.3. Perspectivas particulares[42]

En este apartado recogemos algunas propuestas que relacionan la inteligencia artificial distribuida con otros campos: los sistemas de información abiertos, los ecosistemas y la interoperabilidad software.

1.1.3.1. Sistemas de información abiertos[43]

Hewitt propone utilizar los sistemas de información abiertos como fundamento de la inteligencia artificial distribuida, ya que integran aspectos de sociología y ciencia de sistemas concurrentes, mientras que la inteligencia artificial clásica (fuerte) tiene su base en la neurofisiología, psicología y ciencia cognitiva. Define los sistemas abiertos como los sistemas que están sujetos a eventos imprevistos y que reciben información de otros sistemas en cualquier instante; mientras que los sistemas de información abiertos son sistemas abiertos que emplean almacenamiento digital y tecnología de comunicaciones.

---

42 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 28.

43 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 28.

Se caracterizan por mantener inconsistencias en su conocimiento, operar de forma asíncrona, ser concurrentes, mantener un control descentralizado y mantener conocimiento de otros sistemas, como por ejemplo, las oficinas, transacciones interbancarias, bases de datos distribuidas, etc.

1.1.3.2. Ecosistemas[44]

Los ecosistemas han sido propuestos como una forma de evaluar la dinámica global de los sistemas distribuidos. Se basan en modelar las sociedades de agentes siguiendo una analogía ecológica y ver cómo evoluciona la sociedad con el transcurso de las interacciones.

1.1.3.2.1. Ingeniería software basada en agentes[45]

Los agentes encapsulan los programas y mediante la definición de unas primitivas, permiten el intercambio de órdenes y datos entre los programas. La principal diferencia del enfoque de agentes es la flexibibilidad de la comunicación, que se basa en la existencia de una ontología compartida por las aplicaciones definidas en un lenguaje de representación del conocimiento denominado KIF (Formato de Intercambio del Conocimiento; *Knowledge Interchange Format*) y un lenguaje y protocolo para definir las primitivas de comunicación denominado KQML (Lenguaje de manipulación y consulta de conocimiento; *Knowledge Query and Manipulation Language*).

44 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 28.

45 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 29.

1.1.3.3. Perspectiva del diseñador[46]

La perspectiva del diseñador pretende aclarar qué técnicas y entornos están disponibles para realizar aplicaciones basadas en agentes.

1.1.3.3.1. Metodologías orientadas a agente[47]

Una vez que ha comenzado a establecerse la tecnología de agentes, y se han desarrollado diversas plataformas y lenguajes para emplear sistemas multiagentes en variadas aplicaciones, han comenzado a surgir metodologías que tratan de asistir en el ciclo de vida de la construcción de los sistemas multiagente para obtener las ventajas de reciclaje de los sistemas y mantenimiento, entre otras. Podemos distinguir los siguientes enfoques:

- Metodologías orientadas a agente basadas en metodologías orientadas a objeto: parten de las metodologías orientadas a objeto añadiendo las peculiaridades de los agentes: creencias, objetivos, planes, cómo identificar agentes, relaciones e interacciones entre agentes, etc.
- Metodologías orientadas a agente basadas en metodologías de ingeniería del conocimiento: extienden metodologías de ingeniería del conocimiento, añadiendo principalmente el modelado de las interacciones y la conducta proactiva de los agentes.

46 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 29.

47 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 29.

1.1.3.3.2. Entornos de desarrollo[48]

Los diseñadores, además de conocer las metodologías y lenguajes disponibles, necesitan saber qué herramientas o entornos permiten programar agentes. Tradicionalmente estos entornos han tenido como soporte arquitecturas de pizarra o entornos orientados a objetos. Podemos distinguir dos tipos de entornos:

- Entornos de desarrollo: proporcionan funcionalidades básicas para programar agentes: definición de agentes, funciones para paso de mensajes, un lenguaje de programación de los agentes, un entorno de depuración o visualización. Pueden soportar diferentes arquitecturas de agentes (denominados entornos integradores) o, más frecuentemente, sólo un tipo de agente básico del que se derivan el resto de agentes. Dentro de estos entornos podemos citar a Archon, Imagine, GRATE*, Coopera, MSM, Interrap o Telescript.
- Entornos de experimentación: los entornos de experimentación ofrecen un entorno de modelado de dominios para simular y evaluar teorías sobre agentes. Los entornos más conocidos son DVMT (supervisión de vehículos que emiten señales e interpretación correcta de estas señales), MACE (domino genérico), MICE (para desarrollar un dominio de tipo tablero), Tileworld (simulación de robots en un dominio cambiante, cuyo objetivo es rellenar huecos), Phoenix (simulación del parque Yellowstone, con agentes bomberos), AGenDA (entorno genérico en el que se han modelado compañías de transportes cooperativas y robots autónomos en un muelle de carga) o JavaBots (equipos de fútbol de agentes que aprende con cada gol).

48 Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 29.

2. ONTOLOGIAS WEB

Los agentes encapsulan los programas mediante la definición de unas primitivas, permiten el intercambio de ordenes y datos entre los programa[49].

2.1. Web semántica[50]

La enorme revolución que supuso el uso generalizado de internet impulsó el intercambio de información personal, académica y comercial. La web está a punto de sufrir un nuevo cambio: la información que aparece en Internet va a poder interpretarse por los ordenadores sin necesidad de intervención humana, es la denominada web semántica. Para que esto ocurra, es necesario que la información de las páginas web se codifique mediante ontologías. Las ontologías representarán el conocimiento de Internet, definiendo formalmente los conceptos de los diferentes dominios y sus relaciones, con capacidad para realizar deducciones con este conocimiento.

2.1.1. Descripción[51]

Actualmente, la web es un espacio preparado para el intercambio de información diseñado para el consumo humano. Las páginas web son creadas por personas para ser entendidas por personas. No existe un formato común para mostrar la información, por lo cual, los desarrolladores de páginas web crean sus páginas dependiendo de los potenciales usuarios que van a visitarlas. Los actuales browsers de web realizan la búsqueda de información, con más o menos fortuna, mediante palabras clave que aparecerán en el código HTML[52] de las páginas web dispersas en Internet.

---

49 Iglesias Fernández, Carlos, "Fundamentos de agentes inteligentes", http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps, página 29.

50 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 1.

51 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 1.

52 HTML [Hypertext Markup Language] es el lenguaje de marcado que se utiliza para codificar el formato de presentación y enlaces de hipertexto de las páginas Web

Por otro lado, los agentes de búsqueda actuales no se diseñan para "comprender" la información que reside en la web, precisamente porque es prácticamente imposible conocer la representación de los datos ubicados en las diferentes páginas[53]. La información que puede encontrar el buscador puede estar relacionada con la prensa, demandas judiciales, información que reúna compañías de seguros, etc. Normalmente gastamos mucho tiempo en seleccionar la información que nos puede ser útil, y navegando por las referencias URL[54] hasta encontrar, con suerte, lo que estamos buscando.

Es indudable que las ventajas que ofrece Internet son enormes a la hora de buscar información, pero adolece de una manera de encontrar información de forma precisa y de poder realizar deducciones con la información existente.

2.1.2. El Intercambio de donocimientos en la web semántica

En los últimos años, muchos investigadores están diseñando modelos para transformar la red desde un espacio de información a un espacio de conocimientos. Recientemente, Tim Berners-Lee, uno de los inventores de la web, defiende el desarrollo de la Web con conocimientos, y organizaciones como SematicWeb se encargan de estandarizar lenguajes y herramientas para hacer efectiva la web semántica.

La idea es que los datos puedan ser utilizados y "comprendidos" por los ordenadores sin necesidad de supervisión humana, de forma que los agentes web puedan ser diseñados para tratar la información situada en las páginas web de manera semiautomática.

---

53 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 1.

54 URL (Uniform Resource Locator), lo que todos los lectores conocerán como la dirección de una página Web

Se trata de convertir la información en conocimiento, referenciando datos dentro de las páginas web a metadatos con un esquema común consensuado sobre algún dominio. Los metadatos no sólo especificarán el esquema de datos que debe aparecer en cada instancia, sino que además podrán tener información adicional de cómo hacer deducciones con ellos, es decir, axiomas que podrán aplicarse en los diferentes dominios que trate el conocimiento almacenado[55].

Con ello, se mejorará la búsqueda de información y se potenciará el desarrollo de aplicaciones de comercio electrónico, ya que las anotaciones de información seguirán un esquema común, y los buscadores web compartirán con las anotaciones web los mismos esquemas. Empresas que traten con clientes y proveedores, podrán intercambiar sus datos de productos siguiendo estos esquemas comunes consensuados[56].

Los agentes web no sólo encontrarán la información de forma precisa, si no que podrán realizar inferencias automáticamente buscando información relacionada con la que se encuentra situada en las páginas, y con los requerimientos de la consulta indicada por el usuario.

2.1.3. Las ontologías como soporte de la web semántica[57]

Para que esto pueda llevarse a cabo, se necesita que el conocimiento de la web esté representado de forma que sea legible por los ordenadores, esté consensuado, y sea reutilizable. Las ontologías proporcionan la vía para representar este conocimiento. El término ontología proviene de la filosofía; pero en inteligencia artificial, tiene diferentes connotaciones.

---

55 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 2.

56 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 2.

57 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 2.

La definición declarativa más consolidada la describe como "una especificación explícita y formal sobre una conceptualización compartida". La interpretación de esta definición es que las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada; y que esta conceptualización debe ser representada de una manera formal, legible y utilizable por los ordenadores.

Las ontologías tienen los siguientes componentes que servirán para representar el conocimiento de algún dominio[58]:

- Conceptos: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc[59].
- Relaciones: representan la interacción y enlace entre los conceptos del dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc[60].
- Funciones: son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como categorizar-clase, asignarfecha, etc[61].
- Instancias: se utilizan para representar objetos determinados de un concepto[62].
- Axiomas: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B", "Para todo A que cumpla la condición C1, A es B", etc.

58 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 3
59 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 3
60 Aolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 3
61 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 3
62 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 3

Estos últimos componentes, los axiomas, permiten junto con la herencia de conceptos, inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos. Por ejemplo, con el conocimiento anotado en las páginas web mediante ontologías, podremos utilizar un agente web al que podamos preguntar sobre los dentistas que se encuentren a una cierta distancia de mi casa. Una de las posibles respuestas que me podría ofrecer el agente sería[63]:

* Dra. Macías – Dentista - ortodoncia
- Consulta Particular (85$)- c/Rosa 4 - a 2 km
- Hospital Clínico "La Paz"- c/Principal 17 - a 5 km.

Incluso si la página web de la Dra. Macías no tuviera especificada la dirección del hospital, el agente web de búsqueda podía utilizar un atributo de los dentistas (Trabaja_en_Hospital), y con el valor de este atributo encontrar el atributo Dirección del concepto Hospital y mostrar esta información del hospital concreto de esa instancia de Dentista[64].

Además de utilizar estas propiedades de relaciones y herencia de los conceptos especificados, mediante los axiomas se tendrían mayor capacidad expresiva del dominio almacenado. Por ejemplo, si en este dominio tenemos declarado el axioma[65]:
"Si el médico trabaja en un hospital de mi seguro no tendré que pagar minuta."

El agente podría, utilizando el conocimiento representado en los conceptos, sus relaciones y utilizando el axioma, aconsejarnos sobre los dentistas que cumplieran este requisito.

63 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 3
64 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 3
65 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 3

2.1.4. Cómo alcanzar la web semántica[66]

Para poder explotar la web semántica, se necesitan lenguajes de marcado apropiados que representen el conocimiento de las ontologías. Actualmente, mediante anotaciones RDF-RDF Schema se pueden representar algunas facetas sobre conceptos de un dominio y permite, mediante relaciones taxonómicas, crear una jerarquía de conceptos. Pero se necesitan lenguajes de marcado (basados en RDF) con mayor expresividad y capacidad de razonamiento para representar los conocimientos que contienen las ontologías. De esta forma, existen ya disponibles herramientas como Protégé, OntoEdit, ó WebOnto para realizar anotaciones en páginas web con lenguajes de marcado propios.

El lenguaje con gran capacidad expresiva que está emergiendo como un estándar para realizar anotaciones de ontologías en web es DAML 5, aunque en este momento no tiene sus formatos totalmente definidos. Por otro lado, se necesitan agentes y aplicaciones web que exploten este conocimiento anotado en las páginas web[67].

Estos agentes de conocimientos web serán capaces de interpretar los esquemas ontológicos y axiomas de diferentes dominios, mantendrán la consistencia de las instancias que se inserten en las páginas web siguiendo los esquemas ontológicos definidos, realizarán una búsqueda con inferencias utilizando los axiomas situados en los esquemas, y podrán realizar ligaduras de los árboles taxonómicos de varias ontologías[68].

66 Adolfo Lozano Tello, “Ontologías en la Web Semántica”, http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 3
67 Adolfo Lozano Tello, “Ontologías en la Web Semántica”, http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 4
68 Adolfo Lozano Tello, “Ontologías en la Web Semántica”, http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 4

Para potenciar el uso de ontologías en la web, se necesitan aplicaciones específicas de búsqueda de ontologías, como (Onto) Agent, que indiquen a los usuarios las ontologías existentes y sus características para poder utilizarlas en su sistema[69]. La web semántica un salto cualitativo sobre el potencial de la Web. Las principales ventajas de esta nueva revolución en Internet serán el desarrollo de aplicaciones con esquemas de datos comunes, fomento de las transacciones entre empresas por comercio electrónico y búsqueda de información con inferencias. Para poder lograr estos objetivos se necesita unificar los contenidos semánticos por medio de ontologías que formalicen este conocimiento de forma consensuada y reutilizable[70].

Se necesita un lenguaje común basado en web, con suficiente capacidad expresiva y de razonamiento para representar la semántica de las ontologías; este hecho parece que lo veremos en muy poco tiempo. Los futuros agentes se diseñarán para explotar el conocimiento de la web[71].

2.2. Ontolgias

2.2.1. Ontologias para la gestion del conocimiento

Una ontología ha de entenderse como un entendimiento común y compartido de un dominio, que puede comunicarse entre científicos y sistemas computacionales, entre personas y sistemas heterogéneos[72]. Ésta última característica, el hecho de que puedan compartirse y reutilizarse en aplicaciones diferentes, explica en parte el gran interés suscitado en los últimos años en la creación e integración de ontologías.

---

69 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 4

70 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 4

71 Adolfo Lozano Tello, "Ontologías en la Web Semántica", http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf, página 4

72 http://es.geocities.com/ontologia04/definicion.htm

Cuando hablamos de ontologías como "sistemas de representación de conocimiento" debemos especificar a qué tipo de sistemas nos referimos. En realidad, las ontologías se están empleando en todo tipo de aplicaciones informáticas en las que sea necesario definir concretamente el conjunto de entidades relevantes en el campo de aplicación determinado, así como las interacciones entre las mismas. Algunas ontologías se crean con el mero objetivo de alcanzar una comprensión del conocimiento pertinente, ya que su creación impone una especificación muy detallada[73].

2.2.2. Tipologías

Ontologías de un dominio, en las que se representa el conocimiento especializado pertinente de un dominio o subdominio, como la medicina, las aplicaciones militares, la cardiología o, en nuestro caso particular, la oncología.
Ontologías genéricas, en las que se representan conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos o los tipos de objetos.

Ontologías representacionales, en las que se especifican las conceptualizaciones que subyacen a los formalismos de representación del conocimiento, por lo que también se denominan meta-ontologías (meta-level o top-level ontologies).

Ontologías creadas para una actividad o tarea específica (denominadas task ontologies), como por ejemplo la venta de productos o el diagnóstico de una enfermedad y las ontologías creadas para una aplicación específica.

73 Chantal Pérez, http://elies.rediris.es/elies18/531.html.

2.2.3. Tipos de ontologías

Tipos de ontologías de acuerdo a su alcance de aplicabilidad[74]:

1. Ontología de la aplicación: usadas por la aplicación. Ontología de procesos de producción, de diagnóstico de fallas, de diseño intermedio de barcos, etc.
2. Ontología del dominio: específicas para un tipo de artefacto, generalizaciones sobre tareas específicas en algún dominio. Por ejemplo, ontología del proceso de producción de hidrocarburos, de la red eléctrica, de barcos, etc.
3. Ontologías técnicas básicas: describe características generales de artefactos. Por ejemplo: componentes, procesos, funciones.
4. Ontologías genéricas: describe la categoría de más alto nivel.

Otra forma de indentificar ontologías es desde su punto de vista: por ejemplo: físico, de comportamiento, funcional, estructural, topológico, etc.

Otra forma de indentificar ontologias es desde su punto de visa: por ejemplo: fisico, de comportamiento, funcional, estructural, topologico, etc.

2.2.4. Diseño de ontologías[75]

Cuando decidimos cómo representar algo en una ontología estamos haciendo decisiones de diseño.

1. Claridad: una ontología debe de poder comunicar de manera efectiva el significado de sus términos. Las definiciones deben de ser objetivas y comentadas en lenguaje natural.

74 Eduardo Morales, http://ccc.inaoep.mx/~emorales/Cursos/RdeC/node197.html.

75 Eduardo Morales, http://ccc.inaoep.mx/~emorales/Cursos/RdeC/node198.html

2. Coherencia: debe de permitir hacer inferencias que sean consistentes con las definiciones.
3. Extendible: debe de anticipar usos y permitir extensiones y especializaciones monotónicas.
4. Sesgo de codificación mínimo (Minimal encoding bias): debe de especificar al nivel de conocimiento sin depender de una codificación particular a nivel de símbolo.
5. Mínimo compromiso ontológico: debe de hacer la menor cantidad de ”pretensiones” acerca del mundo modelado. En estos criterios de decisión se tienen que hacer balances.

En estos criterios de decisión se tienen que hacer balances.

2.2.5. Metodología de construcción de ontologías[76]

1. Especificar el contexto de aplicación y el punto de vista del modelado. El contexto de aplicación describe el dominio de aplicación, los objetos de interés del dominio y las tareas que se van a realizar por la ontología (para que se va a construir). El modelado del punto de vista describe el tipo de modelo, tales como, dinámico - estático, funcional - causal, etc.
   Por ejemplo: dentro de la electricidad podemos pensar en 4 grandes conceptos: generación, distribución, transporte y consumo.
   Los componentes de la funcionalidad son: generadores, líneas de transmisión, capacitores, transformadores, cargas, etc.
   Algunas variables son: voltage, intensidad, potencia, y sus leyes.

2. Hacer un diseño preliminar basándose en una ontología existente.
   Implica una etapa de análisis y de mapeo de ontologías

76 Eduardo Morales,http://ccc.inaoep.mx/~emorales/Cursos/RdeC/node200.html

El mapeo puede ser:

- de formalización: de la especificación o del modelado.
- para aumentar la parte declarativa de la ontología.
- para especializar términos creando subtipos o restricciones de tipos.
- mezcla de todos

Este paso es el más difícil e implica mayor trabajo.

Por ejemplo, una línea de transmisión es un elemento que transporta energía eléctrica, genera pérdidas y baja el voltage. El proceso de transporte de energía es caso específo de un proceso físico, por lo que podemos tomar una ontología de procesos físicos. Por otro lado, la descomposición de procesos es otro aspecto que podemos incorporar usando una ontología de descomposición. Después tenemos que verificar que los conceptos encontrados en las ontologías son adecuados para nuestro propósito.

3. Hacer un diseño definitivo y evaluarlo

   Básicamente debemos de considerar que la ontología construida va a ser reutilizada. Algunos principios generales de reutilización son: abstracción (lo más abstracto posible, pero suficientemente concreto) modularización (aislar conceptos), jerarquización (orden) y estandarización.

4. Documentación y reutilización.

   La documentación tiene que hacerse en forma paralela a los puntos anteriores y debe de tener el tipo de mapeo en que se basa la nueva teoría, diferencias semánticas con las ontologías seleccionadas, justificación de las decisiones tomadas, evaluación, conocimiento adicional para usarla, etc.

También debe de ser indexada y colocada (ordenada) con las ontologías existentes para su reutilización.

2.3. Implementación de ontologías

La especificación de cualquier ontología necesita de algún lenguaje de representación. Estos lenguajes sirven para definir términos y señalar las relaciones que guardan entre ellos. De lenguajes de representación hay muchos. Algunos como KIF-based Ontololingua, Loom, Frame-Logic basan su expresividad en la lógica de primer orden. Sin embargo, para aplicaciones en la web es importante poseer un lenguaje con sintaxis estandarizada. Debido a que XML [*Extensible Markup Language*] se ha convertido en un lenguaje estándar para el intercambio de datos en la web, se ha hecho común el intercambio de ontologías expresadas en sintaxis XML. Este requisito ha llevado a usar lenguajes basados en sintaxis XML. Algunos de ellos son RDF [*Resource Description FrameWork*], OIL [*Ontology Interchange Language*] y DAML+OIL. RDF tiene grandes garantías de convertirse en un estándar aceptado. DAML + OIL, desarrollado por un grupo de científicos europeos y americanos, se ha convertido en el sucesor de OIL. Sin embargo, antes de empezar hablar de todos estos lenguajes de representación, sería necesario dar una breve descripción sobre que es XML.

XML [*Extensible Markup Language*] es un lenguaje marcado, basado en una sintaxis similar a HTML, que permite a los usuarios definir sus propias etiquetas y estructuras de documentos. El potencial que ofrece XML es utilizado para el intercambio de datos, la creación de protocolos e infraestructuras para registros de empresas (ebXML y UDDI), y la adaptación de interfaces de presentación a múltiples dispositivos (WAP, PDA etc.).

Actualmente, organizaciones y empresas cooperan en la definición de conjuntos de etiquetas para sectores de la industria específicos (turismo, banca, transporte, medicina, etc.), coordinados por el organismo sin ánimo de lucro OASIS.

A pesar de todo, XML, adolece de ciertas deficiencias. XML permite a los usuarios definir etiquetas como por ejemplo, <AUTHOR>, las cuales parecen estar dotadas de cierto significado. Sin embargo, des del punto de vista computacional etiquetas como <AUTHOR> tienen el mismo significado que una etiqueta como <H1>. Un computador simplemente no conoce, que es un autor, ni si el concepto está relacionado con una persona por ejemplo. XML puede ayudar a los humanos a predecir que información puede estar contenida entre etiquetas como en el caso de <AUTHOR></AUTHOR>, pero sólo puede ayudar. Para un procesador XML, <AUTHOR> y <BOOK> son totalmente iguales, no tienen ningún significado.

2.3.1. XML

XML se diseñó como consecuencia del gran crecimiento de Internet, los intereses comerciales y la necesidad de poder realizar páginas Web vistosas. En aquel momento, HTML se había convertido en el mecanismo de presentación de documentos más extendido de la historia. Todas las grandes firmas se anunciaban en Internet a través de páginas HTML y su número no paraba de crecer y crecer. Sin embargo, todo este éxito desembocó en una paradoja. Produjo que evolucionará de forma muy rápida y no siempre tomando el camino más adecuado. De esta forma, HTML mantuvo su rigidez inicial y se convirtió en un lenguaje limitado.

En un lenguaje de presentación de información directamente para humanos, pero difícil de procesar por programas informáticos. HTML permitía indicar si un tipo de letra era azul o si era de un tamaño determinado, pero era incapaz de decir si lo que estaba mostrando era el título de un libro o el precio de un artículo. Por este motivo nació XML. XML permitió describir aquello que etiquetaba.

2.3.2. RDF

2.3.2.1. Definición de RDF

RDF significa "Resource Description Framework". RDF está pensado para la web, RDF es el esfuerzo por identificar los comunes denominadores y proveer un mecanismo para que los webmasters puedan proveer metadatos útiles para todos, sin la necesidad de ninguna intervención humana.

2.3.2.2. Descripción de RDF

EL modelo de datos RDF se basa en tres tipos de objetos:

- Recursos: cualquier cosa que puede ser descrita usando expresiones RDF. Esto es, desde una página web hasta objetos no accesibles por la red, como por ejemplo un simple libro impreso. Los recursos se identifican mediante URIs[77].

77 URI (Universal Resource Identifier) define o especifica una entidad (y puede ser cualquier cosa: documento, dispositivo, vídeo, audio, imagen). Las URL (Uniform Resource Locator), es el tipo más familiar de URI.

- Propiedades: son los atributos que describen a los recursos. Cada propiedad describe los tipos de recursos que puede describir y el rango de valores que puede tomar. Ejemplo: los atributos título o autor pueden ser propiedades del recurso libro impreso. Las propiedades pueden tomar como valores cualquier atómico (cadena de texto, número...) o recurso.
- Sentencias: ternas constituidas por un recurso, una propiedad aplicable a este recurso y el valor correspondiente a dicha propiedad. Dicho de otro modo, las sentencias RDF son simples registros de tres campos (recurso, propiedad y valor), con lo cuál son fáciles de manejar y de usar para buscar información en grandes volúmenes de información como la web [escalabilidad].

El lenguaje utilizado es XML, pero podrían usarse otros lenguajes. La sintaxis RDF/ XML se ha diseñado para permitir agrupar múltiples sentencias sobre un mismo recurso. El elemento Description delimita esta agrupación. Description denomina un recurso, o bien, por medio de un atributo about, o bien, por medio de un atributo ID. Este último, es usado para subrayar que el recurso aún no existe, mientras que about manifiesta todo lo contrario.

Las sentencias contenidas dentro Description son siempre propiedades. Incluso puede haber varias propiedades con el mismo nombre (cada propiedad añade un nuevo arco al grafo). En tal caso, la interpretación de la representación gráfica se definirá por el creador del esquema.

El nombre de una propiedad puede ser cualquier cadena de texto XML bien formada. Además, se puede asociar a un esquema a partir de la concatenación del namespace del esquema RDF y el nombre de la propiedad. Más adelante, veremos que significa namespace y qué es un esquema RDF. Una propiedad puede tomar como valor un literal (cadena de texto, número, etc.) o un recurso. Para especificar que una propiedad toma como valor otro recurso, se usa el atributo resource. Ejemplo:

```
<rdf:Description about="http://www.w3.org/Home/Lassila">
<s:Creator rdf:resource=”http://www.w3.org/staffId/85740/”>
</rdf:Description>
```

Finalmente, el elemento RDF es simplemente un envoltorio que marca los límites del documento RDF.

2.3.2.2. Vocabularios y esquemas RDF

Un vocabulario RDF se define como el conjunto de propiedades estándares, distribuidas en paquetes para describir distintos tipos de datos, por ejemplo, para libros podría ser: autor, título y fecha[78]. La base de los vocabularios en RDF son los esquemas RDF. Estos nacieron como mecanismo para asegurar que tanto lector como escritor de una sentencia [declaración] entendiera el mismo significado, sin ambigüedad, para los términos utilizados, tales como Creator. Puesto que especialmente, en el ámbito global del World Wide Web conviene ser los más preciso posible, el contenido de los esquemas RDF sirve para expresar el significado de los términos utilizados.

Un esquema RDF podemos considerar, a su vez, que es como una especie de diccionario. Es decir, un diccionario que define todos aquellos términos que se utilizarán en una sentencia RDF común y que servirán, a su vez, para dotarla de significado específico.

78Juan Carlos Cámara, http://www.iua.upf.es/~jblat/material/doctorat/students/jccbis/Tecnologias_RDF5.htm

Sin embargo, un vocabulario RDF no es un conjunto de esquemas RDF, compuestos de elementos, valores de estos elementos y sus relaciones semánticas. Es más bien, un conjunto de recursos que evoluciona a través del tiempo. Son los esquemas RDF los que designan versiones particulares de un vocabulario RDF en cualquier momento puntual. Ello nos indica como RDF permite implementar el concepto de evolución del Web Semántico.

En un esquema RDF se documentan las definiciones y restricciones de uso de las propiedades. Para evitar confusiones entre definiciones independientes del mismo término (en otros esquemas), RDF utiliza los namespaces de XML. Los namespaces ["espacios de nombre"] son una forma de asociar el uso específico de una palabra en el contexto de un esquema [diccionario] en el que se puede encontrar una definición determinada.

Es la forma de identificar el uso de una palabra dentro de un determinado contexto. Es decir, si dos esquemas definiesen el término Creator con significados distintos, el uso de esta propiedad en una sentencia podría llegar a generar ambigüedad.

Para solucionar este evento, se asocia el namespace de cada esquema (el w3c declara que cada esquema tiene su namespace propio) a cada propiedad y evitar así confusiones. Los namespaces se expresan en RDF mediante el atributo xmlns. En definitiva los namespaces proporcionan un mecanismo que permite:

- Que las referencias al nombre de una propiedad no sean ambiguas.
- Que los documentos RDF puedan tener información definida por propiedades de distintos esquemas.

En RDF los recursos pueden ser instancias de una o más clases. Las clases se organizan en forma de jerarquías de clases, en las cuales por ejemplo, 'animal' es la superclase de 'persona'. El concepto de clase se corresponde al concepto genérico de tipo o categoría, equivalente al concepto de clase de programación orientada a objetos. Para expresar las relaciones que hay entre las subclases, en RDF, se utiliza la propiedad rdfs:subClassOf.

La propiedad rdf:type se usa para indicar si un recurso es una clase [rdfs:Class], una propiedad [rdf:Property] o simplemente un recurso [rdfs:Resource]. La reunión de rdfs:Class, rdf:Property y rdfs:Resource constituye el corazón de la especificación del RDF esquema. Para entender estos conceptos, veamos un ejemplo:

```
<rdf:RDF xml:lang="en"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<rdf:Description rdf:ID="Animal">
<rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Resource"/>
</rdf:Description>
<rdf:Description rdf:ID="Person">
<rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs:subClassOf rdf:resource="#Person"/>
</rdf:Description>
</rdf:RDF>
```

La ventaja más importante de RDF comparado con otros enfoques de metadatos es que usando RDF, puede decirse cualquier cosa sobre cualquier cosa. Cualquiera puede hacer declaraciones RDF sobre cualquier recurso identificable. Utilizando RDF, los problemas de ampliar metadatos y combinar metadatos de diferentes esquemas desparecen, pues RDF no utiliza documentos cerrados.

2.3.2.3 Características de RDF[79]

RDF esta cuidadosamente diseñado para tener las siguientes características:

- Independencia: Dado que una propiedad es un recurso, toda organización independiente o incluso cada persona puede inventarlas. Podemos inventar una propiedad llamada "Autor" y otros pueden inventar una propiedad llamada "Director" que podría aplicarse, por ejemplo, a recursos asociados con películas. Esta libertad es necesaria porque no hay un dios que se encargue de definir cuáles son las propiedades a usar en todo el mundo
- Intercambio: Dado que las sentencias RDF se escriben en XML pueden ser fácilmente usadas para intercambiar información. Esto eventualmente es necesario aun con la existencia de un dios.
- Escalabilidad: Las sentencias RDF son simples registros con tres campos (Recurso, propiedad, valor), por lo que son fáciles de manejar y de usar para buscar objetos aun en volúmenes realmente grandes. La web ya es lo suficientemente grande y continúa creciendo. Es probable que tengamos en algún momento miles de millones de RDFs flotando a nuestro alrededor algún día. Por eso la escalabilidad es importante.

79 Joshua Tauberer, http://www.xml.com/pub/a/2001/01/24/rdf.html

3. TÍPOLOGIA, CLASIFICACIÓN Y TAXONOMÍA DE AGENTES

Definición de agente.

Un agente es una entidad que reside en un entornor, donde interpreta datos de sensores reflejando eventos del entorno, y ejecuta comandos que provocan cambios en éste. Un agente puede ser software o hardware. En este último caso, se necesita de mucho software para hacer al agente hardware.

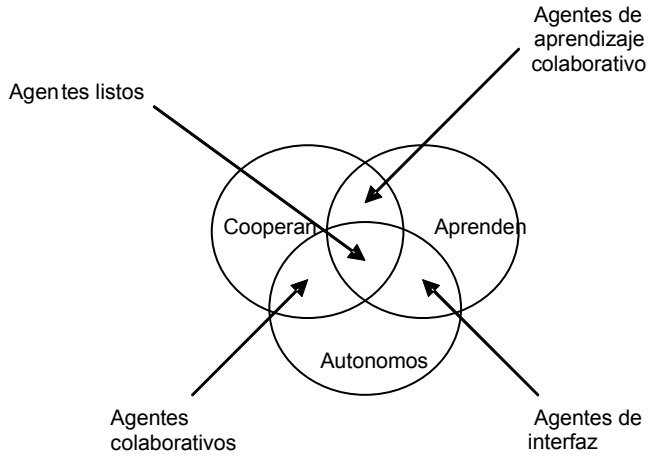
Los agentes habitualmente se definen a partir de un conjunto de atributos o propiedades. Es decir, aquellas propiedades que convierten un sistema hardware y/o software corriente en un agente.

3.1. Tipología

Clasificacion de agentes por sus atributos primarios:
Los agentes se pueden clasificar a lo largo de varios atributos primarios que los agentes deben exhibir como mínimo autonomía, aprendizaje y cooperación.

- La autonomía se refiere al principio de que los agentes pueden operar por ellos mismos sin intervención humana o de otros agentes.
- La cooperación es la razón para tener múltiples agentes y para que ellos cooperen es necesario que los agentes posean una habilidad social, por ejemplo la de interactuar con otros agentes y posiblemente con humanos a través de algún lenguaje de comunicación.
- Finalmente, los agentes son bastante listos, ellos pueden aprender como reacccionar y/o interactuar con su entorno externo.

Figura 8. Tipología de los agentes



Los agentes cooperativos enfatizan más en la cooperación y en la autonomía que en el aprendizaje, pero esto no implica que nunca aprendan. Por su parte los agentes de interfaz hacen más énfasis en la autonomía y en el aprendizaje que en la cooperación.

Además, los agentes pueden ser clasificados por los roles que desempeñan, por ejemplo agentes de información o agentes internet, en esta categoría los agentes se dedican a explorar Internet ayudando con su actividad a manejar gran cantidad de información pueden ser estáticos, móviles o deliberativos. Finalmente, se puede tener una clase de agentes híbridos los cuales combinan dos o más filosofías de agentes en una clase de agente. A partir de la tipología de agentes definida se definirá brevemente sus metáforas esenciales, hipótesis/objetivos, motivaciones, roles, ejemplos prototípicos, beneficios potenciales, cambios claves y algunos otros usos generales acerca de cada tipo particular de agentes.

3.1.1. Agentes colaborativos

Los agentes colaborativos enfatizan en la autonomía y la cooperación para ejecutar tareas por ellos mismos, pueden aprender pero este no es un aspecto típico de su énfasis. Las características generales de estos agentes incluyen autonomía, sociabilidad, responsabilidad y proactividad. Sin embargo ellos deben ser capaces de actuar racional y autónomamente en entornos multi-agente de tiempo comprimido y en entornos abiertos. Tienden a ser estáticos, pueden ser benebolentes, racionales, verdaderos, algunas combinaciones de ellos o ninguna.

3.1.2. Agentes de interfaz

Los agentes de interfaz enfatizan en la autonomía y el aprendizaje para desempeñar tareas para sus propietarios. Esencialmente los agentes de Interfaz soportan y proporcionan asistencia, para que el usuario aprenda a utilizar una aplicación particular tal como un sistema operativo. El agente usuario observa y supervisa las acciones efectuadas por el usuario en la interfaz, aprende nuevos atajos y sugiere mejores formas para realizar las tareas. Así el agente de usuario actúa como un asistente personal autónomo que coopera con el usuario en el cumplimiento de algunas tareas en la aplicación. Los agentes de interfaz aprender a asistir a su usuario de formas diferentes:

- Observando e imitando al usuario
- Por realimentación positiva y negativa desde el usuario (aprendiendo del usuario)
- Recibiendo instrucciones explícitas por parte del usuario
- Solicitando consejo a agentes

La cooperación con otros agentes está limitada a solicitar consejo y no en prologar negociaciones o tratos con ellos, como en el caso de los agentes colaborativos

El objetivo que persiguen los agentes de interfaz es migrar de una metáfora de manipulación directa a una que delegue algunas de las tareas a agentes de interfaz aoftware (proactivo y útil), para acomodar a los usuarios nuevos.

3.3. Agentes móviles

Los Agentes Móviles son procesos software (computacionales) capaces de recorrer o vagar por redes de WAN[80] tales como WWW, interactuando con host extraños, recogiendo información en nombre de su propietario y realizando las obligaciones impuestas por sus usuarios. Esas obligaciones pueden variar desde hacer una reserva de avión a manejar una red de telecomunicaciones. Los agentes móviles son implementaciones de programas remotos es decir programas que se desarrollan en una máquina y se distribuyen en una segunda máquina para su subsecuente ejecución. Además son agentes porque son autónomos y cooperan, a diferencia de los agentes colaborativos, por ejemplo ellos pueden cooperar y comunicarse con otros agentes ubicando algunos de los objetos y métodos internos de otros agentes, de esta manera pueden intercambiar datos e información sin necesidad de dar toda la información.

80 WAN (World Wide Web) Redes de Área Ancha, distancias desde 100 hasta 1,000 km

La utilización de agentes móviles supone los siguientes beneficios:

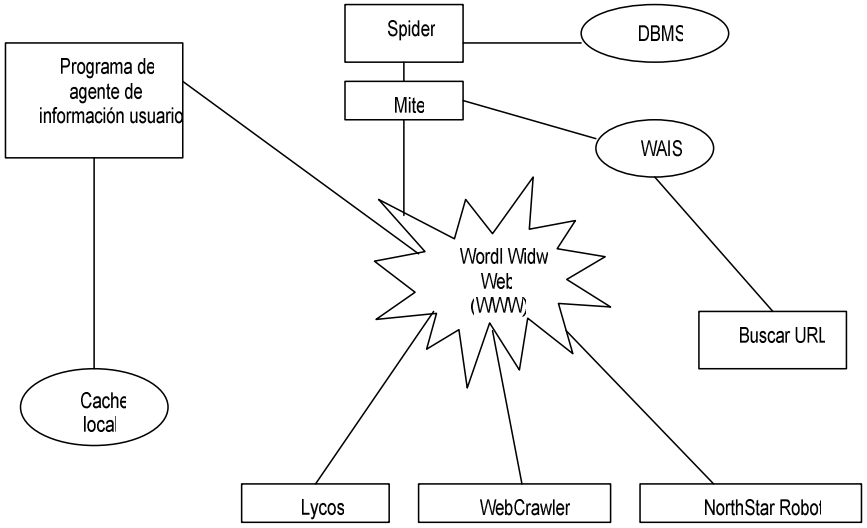
- Reducción en el costo de la comunicación: Hay información que es necesario determinar su relevancia y la transferencia de esta información puede llevar mucho tiempo y congestiona la red.
- Limitación de los recursos locales: El poder de procesamiento y almacenamiento de en una máquina local puede ser muy limitado.
- Coordinación sencilla: Puede ser sencillo coordinar un número de requisitos remotos e independientes y comparar todos los resultados localmente.
- Cálculo asíncrono: que el agente móvil realice tareas y que los resultados se guarden en el buzón de correo por decir algo en algún momento después. Los agentes pueden operar cuando no este el usuario conectado a la red.
- Proporcionan entornos de desarrollo natural para implementar servicios de libre comercio. Estos servicios pueden co-existir con otros de nivel inferior proporcionando al consumidor más alternativas.
- Una arquitectura de cómputo distribuida flexible: los agentes móviles proporcionan una arquitectura de cómputo distribuido única cuyas funciones son diferentes de las asignaciones estáticas permitiendo una forma de nueva de hacer computación distribuida.
- Los agentes móviles representan una oportunidad para hacer un replanteamiento radical y atractivo de los procesos de diseño en general.

La parte crítica de los agentes móviles la constituye la seguridad, por lo que es necesario que en el sistema opere un firewall que evite que las redes internas sean accedidas desde fuera.

3.1.4. Agentes de información/internet[81]

Los agentes de información surgen de la necesidad de manejar el crecimiento de información que se encuentra en la Internet y poder sacar de ella los mayores beneficios. Los agentes de información se encargan de manejar, manipular y coleccionar información de muchas fuentes distribuidas.
Los agentes de información tienen varias características: pueden ser estáticos o móviles, no cooperativos o sociales y pueden o no aprender, de allí que no haya un modelo estándar que defina su modo de operación.

Figura 9. Funcionamiento de los agentes de información



Los agentes Internet pueden ser móviles, pueden ser capaces de atravesar el WWW y recolectar información. Los agentes de información pueden estar asociados a un índice o índices particulares que son capaces de buscar en el WWW, almacenando la topología del WWW en un sistema de base de datos (DBMS). Otras máquinas buscadoras como Lycos o Webcrawler se pueden utilizar para construir el índice.

81 http://personales.upv.es/ccarrasc/doc/2003-2004/WebSemAg/AGENTES.HTM

El usuario del agente de información que requiere una información con ciertas características solicita varias búsquedas a una o varias máquinas de búsqueda de URLs para encontrar la petición, algunas de esas búsquedas pueden hacerse localmente si tiene un cache local. La información se localiza y se envía al usuario. Los agentes de información son similares a los agentes de interfaz o a los agentes móviles. Si los agentes de información son estáticos, entonces se les aplica los cambios de los agentes de interfaz, sin embargo si son móviles se aplica los cambios de los agentes móviles. De otra parte las dificultades que se presentan en los agentes de información son similares a las presentes en los agentes de interfaz y los móviles dependiendo de si ellos son estáticos o móviles respectivamente.

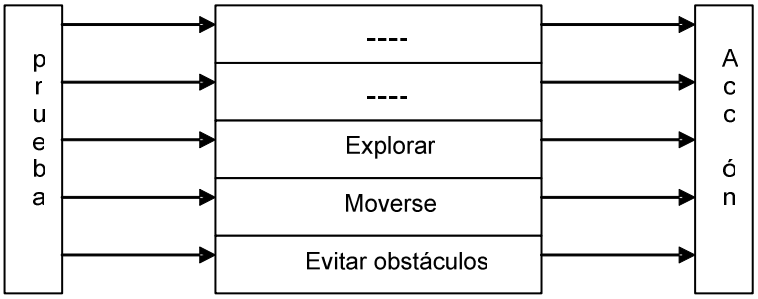
3.1.5. Agentes de software reactivos

Los agentes reactivos representan una categoría especial de agentes que no poseen modelos simbólicos internos de sus entornos, en su lugar reacciona o responde en modo de estímulo respuesta para representar el estado del entorno en el que están empotrados. Algunas de las características soportadas por los agentes reactivos son:

- La funcionalidad: No hay una especificación a priori del comportamiento de los agentes reactivos.
- Descomposición de tareas: Un agente reactivo es visto como un conjunto de módulos que operan autónomamente y que son responsables de tareas especificas, la comunicación entre los módulos es minimizada y de naturaleza de bajo nivel completamente. No existe un modelo global dentro de los agentes y tiene que surgir un comportamiento global.

Los agentes reactivos tienden a operar sobre representaciones que se aproximan a datos en bruto, en contraste a las representaciones simbólicas de alto nivel que está presente en otros tipos de agentes.

Figura 10 Arquitectura de subsunción



Los sistemas de agentes listos (reactivos), se pueden desarrollar a partir de agentes simples que no tienen modelos simbólicos internos y el que seán listos se deriva del comportamiento que surge de las interacciones entre varios módulos. Es importante anotar que los agentes reactivos actuales no poseen necesariamente actuadores y sensores que los conecten al mundo físico. Los agentes reactivos son simples y fáciles de entender y su economía cognitiva es bastante baja, ya que tienen que recordar muy pocas cosas. Ellos no hacen planes hacia delante o revisan algunos modelos del mundo y sus acciones dependen de lo que pasa en el momento presente.

La ventaja de trabajar con agentes reactivos es que son más robustos y tolerantes que otros sistemas basados en agentes, por ejemplo un agente se puede perder, pero sin efectos catastróficos. Otros beneficios que se incluyen son la flexibilidad y la adaptabilidad en contraste con la inflexibilidad, tiempo de respuesta lento y la vulnerabilidad de los sistemas de inteligencia artificial clásicos.

Otro beneficio es que este tipo de trabajo puede direccionar el problema de estructura que ha sido difícilmente abordable por medio de las técnicas tradicionales de inteligencia artificial tales como el razonamiento no monotónico.

Hay un número relativamente pequeño de aplicaciones software basadas en agentes reactivos, por esta razón no hay un modo estándar para su operación, tienden a depender de la arquitectura de un agente reactivo seleccionado. Por ejemplo en el caso de la arquitectura de subsunción de un agente reactivo. Este tipo de arquitectura se ha utilizado para implementar robots físicos. La arquitectura consta e un conjunto de módulos cada uno de los cuales es descrito en un lenguaje de subsunción basado en una máquina de estados finito aumentada (AFSM). Esta máquina se pone en acción si la señal de entrada excede algún valor de umbral, aunque este también depende de los valores de supresión e inhibición de la señal en la AFSM. Las máquinas AFSM representan solamente las unidades procesadas en la arquitectura, por ejemplo no hay símbolos como los que se encuentran en los trabajos de inteligencia artificial. Los módulos se agrupan y se colocan en capaz (las cuales trabajan asícronamente), tal que los módulos en los niveles más altos pueden inhibir esos en capas más bajas. Cada capa tiene un comportamiento, por ejemplo evitar obstáculos o facilitar/controlar el movimiento de un lugar a otro. El área favorita para el desarrollo de aplicaciones es la de los juegos y la industria del entretenimiento.

Los sistemas basados en agentes reactivos se pueden utilizar para simular muchos tipos de mundos artificiales así como también fenómenos naturales, por ejemplo para simular una sociedad de hormigas donde cada hormiga es modelada como un agente.

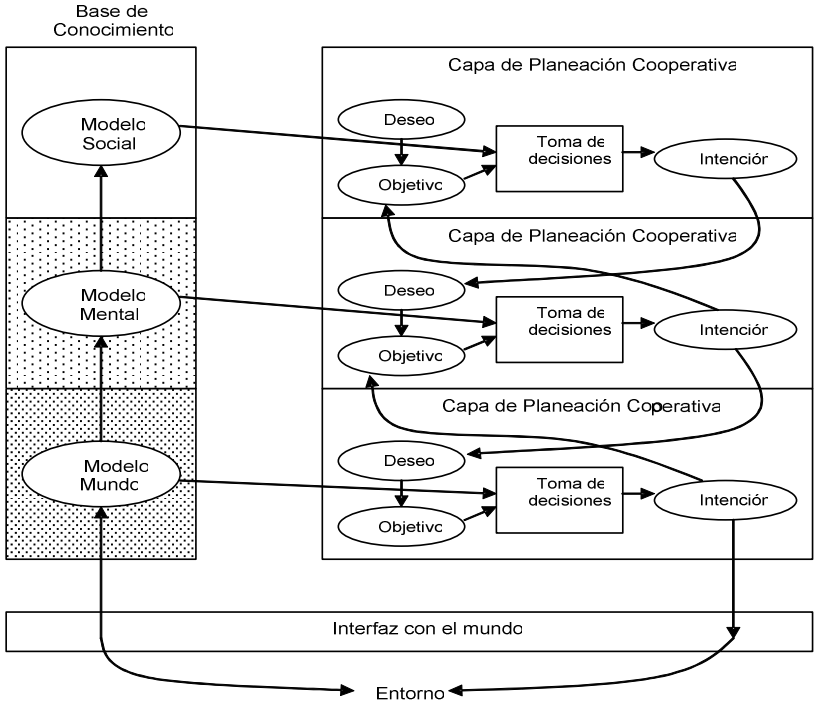
Los agentes reactivos pueden convertir el ordenador en un laboratorio virtual donde el investigador puede modificar algunos parámetros experimentales y validar sus modelos utilizando tanto datos cualitativos como cuantitativos.

3.1.6. Agentes híbridos

Hasta ahora los tipos de agentes discutidos poseen cada uno sus propias fortalezas y deficiencias, obviamente lo que se busca es maximizar las fortalezas y minimizar las deficiencias de las técnicas más relevantes para propósitos particulares. Una forma de hacerlo es adoptar una aproximación híbrida. Los agentes híbridos se refieren a aquellos cuya constitución es una combinación de dos o más filosofías de agentes para formar un agente único.

Para el desarrollo de algunas aplicaciones el tener una arquitectura de agentes híbrida puede ser de ayuda si se quiere tener mayores beneficios de la arquitectura, ya que se aprovecha lo mejor de la filosofía de cada agente, por ejemplo en el caso de agentes reactivos se puede aprovechar la robustez, tiempo de respuesta rápida y adaptabilidad. El problema de estructura es también solventado con la utilización de un componente reactivo. La parte deliverativa de un agente manejan los asuntos orientados a objetivos y los definidos a más largo plazo. La arquitectura de los agentes híbridos (como la InteRRaP desarrollada por el centro de investigación Alemán), implementa una aproximación en capaz para el diseño de agentes. Esta arquitectura se puede utilizar para construir un agente tal como un robot autónomo y consta de una base de conocimiento asociada a una unidad de control. Hay tres capaz de control en la arquitectura: una capa basada en comportamiento, una de planeación local y una de planeación cooperativa.

Figura 11. Agentes híbridos



La arquitectura utiliza las filosofías de agentes reactivos y deliverativos, con la primera se consigue más robustez, eficiencia y reactividad implementadas en la capa basada en el comportamiento la cual contiene un conjunto de patrones de comportamiento (PoBs) y que sirven para describir las habilidades de los agentes reactivos que se implementan en situaciones en las que tienen que responder ante situaciones de tiempo crítico. La capa de planeación local implementa comportamientos orientados a objetivos, mientras que la capa de planeación cooperativa hace posible la planeación y cooperación entre agentes, para realizar tareas y resolver conflictos conjuntamente. Las capaz de planeación y cooperación permiten más deliveración. Cada una de estas capaz esta relacionada con un modelo social, mental y del mundo respectivamente perteneciente a la base del conocimiento.

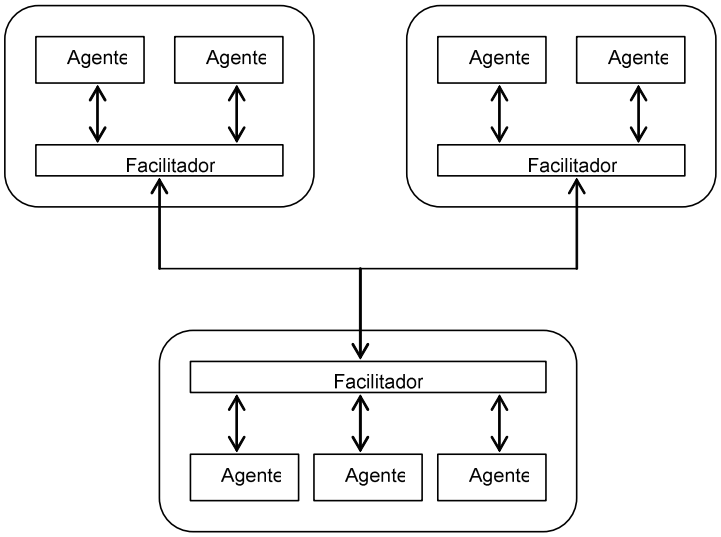
3.1.7. Agentes heterogéneos

Los sistemas de agentes heterogéneos integran diferentes clases de agentes y pueden además contener una o más clases de agentes híbridos.
En el mundo abunda una gran cantidad de productos software que proporcionan una amplia gama de servicios para un amplio rango de dominios. Aunque estos programas trabajan por separado, se ha incrementado la necesidad de lograr que interoperen. Ha surgido un nuevo dominio denominado Ingeniería del software basada en agentes cuya función es facilitar la interoperabilidad entre los diferentes agentes software. Los beneficios de tener una tecnología de agentes heterogéneos son varios:

- Lograr que diferentes aplicaciones interoperen y trabajen de forma cooperativa permite aprovechar las capacidades de cada una.
- Los problemas referentes a la legalidad del software pueden mejorar porque se puede obviar la necesidad de reescribir software costoso, dándole la oportunidad de continuar interoperando con otro sistema.
- La ingeniería basada en agentes software proporciona una nueva aproximación para el diseño, implementación y mantenimiento de software en general y de interoperabilidad del software en particular.

Existen dos arquitecturas posibles, una en la que todos los agentes manejan su propia coordinación y otra en la que los grupos de agentes dependen de otros programas de sistemas para realizar la coordinación. La desventaja es que la comunicación no asegura la dimensionabiliad que es necesaria para el trabajo futuro de los agentes. Se prefiere una aproximación federada.

Figura 12. Agentes federados

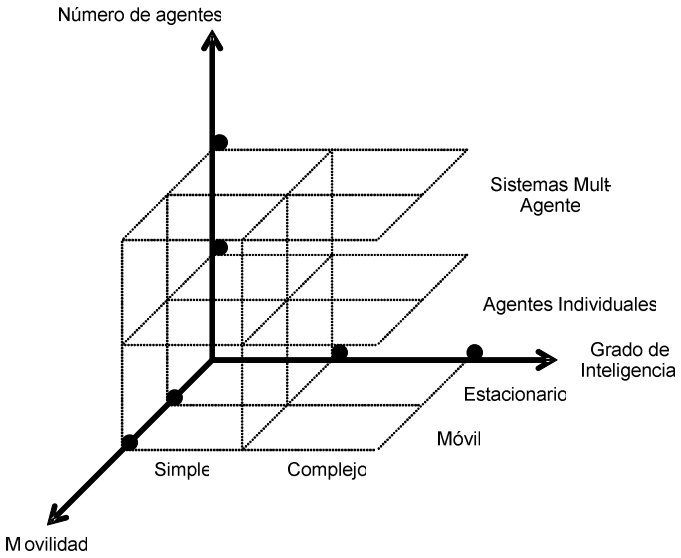


Los agentes federados no se comunican directamente entre sí sino que lo hacen a través de un facilitador o mediador que ubican a otros agentes en la red los cuales son capaces de proporcionar otros servicios. Los sistemas de agentes heterogéneos son una nueva disciplina de los sistemas inteligentes de cooperación e Información en los cuales se integran los sistemas de información, la ingeniería software, las bases de datos y la inteligencia artificial utilizando agentes de información. El trabajo de los sistemas de agentes heterogéneos está progresando y requiere de metodologías, herramientas, técnicas y estándares para lograr la interoperabilidad entre las distintas fuentes de información heterogéneas.

3.2. Clasificación

Todos los tipos de agentes se pueden asignar a una matriz multidimensional construida a partir de las diferentes características, pero en este caso se consideran tres criterios de gran relevancia. Los criterios seleccionados son los adecuados para realizar la clasificación de los actuales y futuros tipos de agentes y permiten además que todas las características mencionadas anteriormente se puedan representar mediante uno o más criterios. Los criterios de los que se habla son Inteligencia, movilidad y número de agentes.

Figura 13. Matriz de clasificación para los sistemas de agentes



Se utilizan los términos simple y complejo para referirse al grado de inteligencia de los agentes.Los agentes simples tienen una cantidad limitada de inteligencia, considerando los sistemas complejos que demuestran un alto grado de comportamiento inteligente. Se pueden diferenciar también dos tipos de agentes móviles dentro del criterio de clasificación de movilidad objetos móviles y scripts móviles.

Los scripts móviles son enviados por enviados a otro ordenador antes de que se ejecuten en ese ordenador. En contraste los objetos móviles pueden cambiar su posición en cualquier momento durante su ejecución, en este caso no solo se transfiere el estado actual del objeto sino también el estado actual y el entorno del sistema.

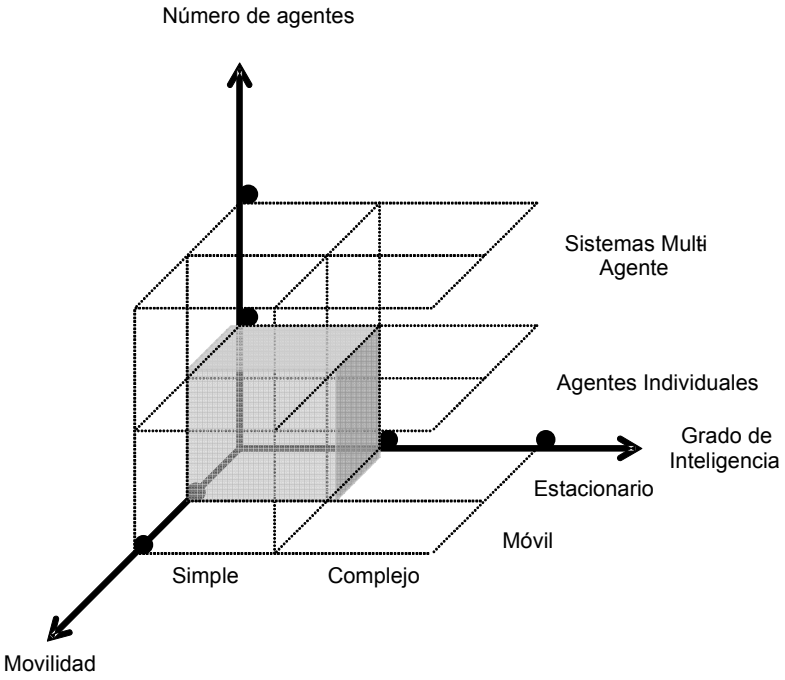
Los agentes individuales se mueven en un entorno que no contiene ningún otro agente, en otras palabras no son capaces de contactar con otras agentes solo contactan con su usuario y con diferentes fuentes de información tales como bases de datos.

En contraste los sistemas multiagente cuentan con numerosos agentes que pueden comunicarse o cooperar con otros.

Todas las características descritas en los párrafos anteriores puedense pueden situar en las tres dimensiones de la matriz: reactividad, proactividad, razonamiento, habilidad para aprender y el carácter son propiedades que determinan la inteligencia de un agente. La capacidad de comunicación está presente tanto en agentes individuales como en los sistemas multiagente y se puede colocar dentro e la categoría de número de agentes. La capacidad de cooperación afecta tanto al criterio de Inteligencia como al del número de agentes. El comportamiento autónomo afecta la Inteligencia y la movilidad de un agente. La operación de un agente móvil es útil solo cuando tiene un máximo grado de autonomía.

En general, las distintas tareas realizadas por los agentes, información, cooperación, transacción, se pueden asignar a los campos de la matriz de clasificación.

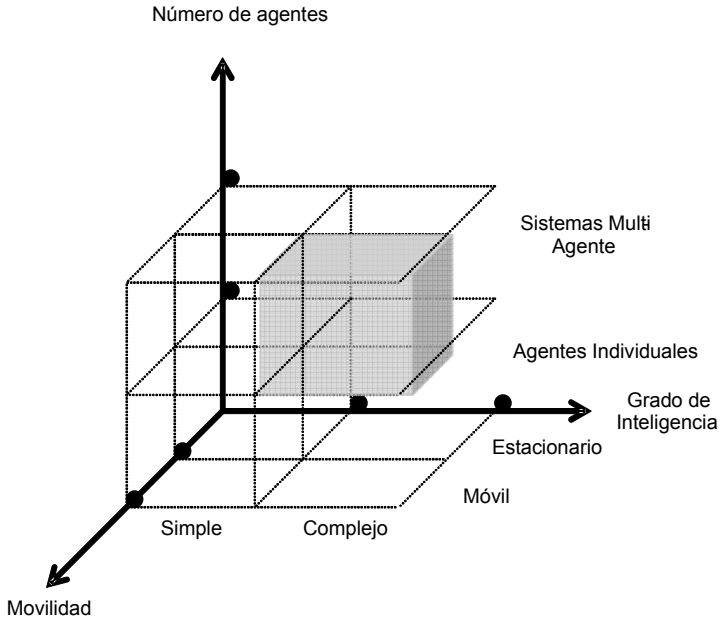
Figura 14. Ubicación de los agentes de información dentro de la matriz de clasificación para los sistemas de agentes



Los agentes de información tienen una cantidad relativamente limitada de inteligencia y pertenecen al área de agentes individuales, además son agentes que operan individualmente porque no necesitan una cooperación extensiva de varios agentes para desempañar las tareas de búsqueda de información. La mayoría de los agentes que existen en la actualidad tienen una naturaleza estacionaria, sin embargo, es especialmente deseable que los agentes de información se realicen como agentes móviles, de esta forma se incrementa la autonomía y se reduce la carga en la red. En consecuencia se espera un incremento en la proporción de agentes móviles cuando se den las condiciones de infraestructura adecuadas para ellos.

Los agentes cooperativos deben tener un alto grado de inteligencia y su ubicación debe corresponder al área de agentes complejos. El segundo criterio de selección se aplica porque ellos deben ser activos en un entorno multiagente. La movilidad no es esencial porque los sistemas de agentes orientados a cooperación se concentran en el proceso actual de solución del problema y se desarrollan y conciben para operar dentro de un sistema de computación. Sin embargo, la movilidad puede ser un criterio deseable para los agentes cooperativos.

Figura 15. Ubicación de los agentes de transacción dentro de la matriz de clasificación de los sistemas de agentes



Los agentes de transacción pueden ser utilizados como agentes individuales y en sistemas multiegante, por ejemplo un número de agentes que realizan transacciones con otros agentes involucrados en la compra y venta electrónica basada en agentes.

De otra parte, los agentes se usan para supervisar transacciones dentro de una red de telecomunicaciones como un sistema individual. Particularmente no se hacen demandas de su inteligencia aún más, la robustez de la arquitectura del agente tiene una importancia relevante. El uso de los agentes de transacción en sistemas multiagente puede ser una excepción. Si por ejemplo las negociaciones complejas tienen lugar entre varios agentes y un agente debe tener una estrategia de negociación clara, luego este es relacionado con las correspondientes demandas de su inteligencia. Los agentes de transacción se pueden realizar como agentes móviles o estacionarios. La decisión depende del escenario de aplicación asociado.

3.3 Taxonomia de los agentes

Los agentes se pueden clasificar dentro de las diferentes tecnologías existentes hoy en día. Se habla entonces de una clasificación dentro del contexto de “Sistemas de Agentes Individuales” donde se encuentran los agentes locales y los agentes de redes y sistemas Multiagentes que contienen también agentes nasados en DAI y agentes móviles.

Agentes individuales

En un sistema de agentes Individuales un agente ejecuta tareas en nombre del usuario o de algún proceso. Mientras ejecuta esas tareas el agente puede comunicarse con el usuario así como también con otros sistemas de recursos remotos o locales. En contraste, los agentes en un sistema multiagente (MAS) puede cooperar extensivamente con cada uno de los demás para realizar sus propios objetivos, por supuesto en esos sistemas el agente puede interactuar con los recursos del sistema y con los usuarios.

La tabla siguiente muestra la clasificación de la tecnología de agentes y sus correspondientes áreas de aplicación:

Tabla I. Clases de agentes inteligentes y sus aplicaciones

Sistemas de Agentes Individuales		Sistemas multiagente	
Agentes locales	Agentes de red	Agentes basados en inteligencia artificial distribuida	Agentes móviles
-Asistentes personales -Asistentes de consejo -Planificador de reuniones	-Correos Listos -Agentes recuperadores de información -Automatización de procesos	Resolución distribuida de problemas	-Telecomunicaciones -Comunicaciones personales -Manejo de redes -Servicios sobre demanda -Mercado Electrónico

3.3.1 Agentes locales

Los miembros de esta clase acceden solamente a recursos locales. Generalmente un agente local actúa como un agente de ayuda (por ejemplo en los sistemas de ayuda inteligentes), o como asistentes personales que soportan a usuarios humanos durante su trabajo diario. El objetivo de estos agentes es colaborar con el usuario por ello el principal énfasis en la investigación lo constituye el campo de interacción usuario/agente. Por otra parte este tipo de agente ha sido llamado también agente de interfaz o interfaz inteligente Los agentes locales pueden asistir al usuario de diferentes formas: ejecutan tareas en nombre del usuario, pueden ocultar la complejidad de tareas difíciles, actuar asíncronamente o aprender y enseñar al usuario. La cantidad de tareas que el usuario puede asistir es virtualmente ilimitada: recuperar y filtrar información local, manejar correo local, planear reuniones, etc.

3.3.2. Agentes de red

En contraste con los agentes locales, los agentes de red pueden acceder no solo a recursos locales sino también a los remotos, y tienen un conocimiento de la infraestructura de la red y de los servicios disponibles dentro de la red. La principal diferencia entre esta clase de agentes y los sistemas multiagente es que los agentes de red no asumen la cooperación con otros agentes. Muchos de los agentes de redes existentes actúan como asistentes personales. Los ejemplos más populares probablemente son los carteros listos y motores de búsqueda. Ellos no solo proporcionan una interfaz inteligente al usuario, sino que también hacen uso extensible de varios servicios disponibles en la red. En contraste con los carteros listos los cuales desarrollan un filtrado avanzado del correo basado en las preferencias del usuario, los motores de búsqueda reúnen el conocimiento disponible en la red en nombre del usuario. Este tipo de agente se conoce también como KnowBot o Softbot.

Cuando reúne la información el agente no solamente oculta la complejidad de la red sino que accede a un amplio rango de información heterogénea y a protocolos invisibles para el usuario y finalmente presenta los resultados de la información al usuario. Este tipo de agente es especialmente importante para Internet. El increíble crecimiento del WWW ha generado una creciente demanda de herramientas para el soporte y manejo de las grandes cantidades de información disponible en la red.

Entre las muchas soluciones desarrolladas hay varias implementaciones basadas en agentes conocidas como WebCrawlers, Spiders y Robots.

3.3.3. Agentes basados en inteligencia artificial distribuida

La principal preocupación en los sistemas multiagente basados en DAI es lo coordinación entre un conjunto de agentes autónomos inteligentes, por ejemplo como ellos coordinan sus objetivos, habilidades, conocimiento, como planifican la realización de tareas o como resuelven problemas. Este tipo de agentes se utiliza en un amplio rango de aplicaciones por ejemplo en la supervisión distribuida de vehículos, la fabricación de computadores integrados, planeación de transporte y en particular en la gestión y manejo de las telecomunicaciones. Estos agentes se desarrollan mediante técnicas de IA, como sistemas basados en reglas, ejemplos basados en razonamiento. Las representaciones aplicadas al conocimiento y los mecanismos de inferencia son básicamente iguales a los utilizados en los sistemas basados en conocimiento (no-distribuidos). En los sistemas multiagente basados en DAI un agente puede comunicarse con el usuario, con los recursos del sistema y con otros agentes y cooperar en la realización de tareas, esta cooperación se establece por medio de un protocolo de contratación que permite la negociación entre agentes.

3.3.4. Agentes móviles

Los agentes móviles están orientados a ofrecer gran número de servicios sofisticados en grandes redes de computadores, por ejemplo para el filtrado avanzado en Internet y agentes de búsqueda, mensajeros listos, conexión de redes, manejo y comunicación inteligente. Los agentes de este tipo se desarrollan por medio de lenguajes scripting por ejemplo TCL, Java, Telescript.

Los dos primeros se utilizan para realizar operaciones de sincronización y ejecución remota de aplicaciones dentro de Internet sin la cooperación de agentes. La metáfora utilizada para la tecnología Telescript es el mercado electrónico. Dentro de este mercado los agentes ejecutan tareas asíncronas en nombre del usuario. Ellos pueden comunicarse con el usuario, con otros servicios disponibles en la red y con otros agentes, sin embargo los mecanismos sofisticados de cooperación entre agentes están aún fuera del alcance de Telescript.

La tecnología existente soporta la movilidad de los agentes, mientras que TCL y Java sólo permiten la ejecución remota de agentes dentro de Internet, los agentes Telescript pueden migrar mientas están activos. En la tabla siguiente resume las propiedades de las clases de agentes identificadas, se debe hacer énfasis en que la taxonomía de los conceptos de IA no es completa y debe ser considerado solamente como un medio para categorizar los conceptos de agentes existentes.

Tabla II. Clases de agentes y sus características

Clases / Atributos	Agentes Locales	Agentes de Red	Agentes basados en DAI	Agentes Móviles
Inteligencia	✓	✓	✓	✓
Operación Asíncrona	✓	✓	✓	✓
Comunicación		✓	✓	✓
Cooperación			✓	✓
Movilidad				✓

La movilidad de los agentes es probablemente la propiedad más cambiante de los agentes y que puede influir en la forma en como se realizan las comunicaciones y los servicios. Sin embargo, la seguridad representa la clave de la movilidad de los agentes. Consecuentemente los agentes se pueden evaluar en entornos de ejecución abiertos y seguros. Los lenguajes scripting se consideran altamente portables y proporcionan un alto grado de seguridad.

La idea de realizar operaciones cliente/servidor mediante la transmisión de programas ejecutables entre clientes y servidores por ejemplo el despacho de programas para ejecución remota es bastante viejo y se denomina procesos de trabajo remoto en los 70's y función de expedición o evaluación remota en los 80's. En el pasado la principal motivación para aplicar este principio fue la falta de capacidad para ejecutar programas localmente y el deseo de compartir recursos y mejorar el balance de carga en los sistemas distribuidos. Estos conceptos se diseñaron para entornos específicos o entornos cerrados, los nuevos conceptos de agentes apuntan a entornos abiertos (por ejemplo dentro de Internet).

La migración de procesos software generalmente se utilizan para conseguir balance en los sistemas de operación distribuidos, pueden considerarse como una extensión del concepto de ejecución remota. Aparte de su código y de los datos un agente puede tener también un estado de ejecución explícito, el cual permite suspender la ejecución de un estado específico en un nodo y regresar después a otro nodo, además este tipo de agente es capaz de crear agentes hijos para distribuir tareas más eficientemente lo que significa que puede realizar tareas en nodos diferentes de manera coordinada.

Hoy en día se puede referir a este tipo de agentes como agentes intinerarios y representan el área más cambiante dentro del campo de los agentes. En principio las siguientes tareas se pueden identificar como emergentes de la tecnología de agentes:

- Procesamiento de tareas asíncrono y cooperativo: la posibilidad de delegar tareas específicas por medio de agentes móviles hacia un nodo específico o a múltiples nodos permitiendo que los cálculos se realicen dinámica y paralelamente. Particularmente soporta la desconexión de tareas y ordenadores clientes débiles.
- Caracterización y configuración de servicios: en comercio electrónico la tecnología de agentes permiten proporcionar servicios por caracterización o (re)configuración de servicios existentes. En este caso los agentes actúan como adaptadores de servicio y se pueden instalar fácilmente.
- Uso de servicio instantáneo y negociación activa: los agentes móviles proporcionan acceso espontáneo a nuevos servicios lo que en el futuro facilitará la distribución de clientes de servicio que se pueden explotar para desarrollar actividades de negociación.
- Descentralización de la gestión: Los agentes móviles permiten disminuir la presión de sistemas de gestión de redes centralizadas y redes de banda ancha por medio de la delegación de tareas específicas de gestión desde el sistema de operación central a agentes de gestión dispersos.

- Comunicaciones inteligentes: Los agentes proporcionan las bases para las comunicaciones avanzadas soportan la configuración de entornos de configuración de usuario donde realizan el control de la entrada y salida de la comunicación en nombre del usuario. Esto incluye exploración de la comunicación, adaptación inteligente de servicios (por ejemplo conversión de formatos de información) para acordar el acceso a la red y a los dispositivos finales, así como también en los servicios avanzados de interconexión e integración de red.
- Almacenamiento información y soporte de tipos de información dinámicos: Los agentes móviles proporcionan un medio efectivo para el almacenamiento de información y servicios dentro de un entorno distribuido y soporte para tipos de información dinámica dentro del correo electrónico y sistemas de redes de información avanzada.

4. APLICACIONES DE AGENTES

Desafortunadamente muchas de las tecnologías existentes son inadecuadas para construir aplicaciones complejas y sistemas distribuidos. Se requieren nuevas herramientas para construir esos sistemas complejos, en ese sentido el software de agentes autonomos inteligentes representa una buena solución para implementar sistemas de procesamiento de información complejos, sin embargo no es apropiado para implementar todas las clases de aplicaciones.

Un software verdaderamente inteligente posee las capacidades del software de agentes (autonomía, comunicabilidad, percepción) y las capacidades del software inteligente (habilidad para explorar el conocimiento, tolerar errores, aprender y razonar en tiempo real y comunicarse en un lenguaje apropiado). Un software con más inteligencia debe poseer más capacidades.

El diseñador del sistema debe utilizar los agentes solo en esas aplicaciones donde pueda obtener un beneficio notable de su uso[82]. Una solución basada en agentes se hace más sensible cuando la aplicación puede explotar las capacidades de la arquitectura básica de agentes, es decir los agentes deben utilizarse en aplicaciones donde una o más de las siguientes condiciones se cumplen:

- Debe operar autónomamente.
- Debe comunicarse con otro agente software o humano.
- Debe supervisar el estado de su entorno y decidir acerca de sus propias actividades basándose en el estado de ese entorno.

82 “Sistemas de Información para el Control de Gestión”, www.cybertesis.cl/tesis/uchile/2005/alvear_t/sources/alvear_t.pdf.

Operación autónoma: con frecuencia el diseñador de software debe construir un sistema que requiere poca intervención humana, algunas veces esta clase de operación autónoma puede obtenerse utilizando scripts, sin embargo una solución solamente basada en scripts no tiene la inferencia requerida en aplicaciones complejas.

Comunicación: los programas de aplicación modernos deben ser capaces de comunicarse con otros programas de aplicación remotos, pero los desarrolladores necesitan formas efectivas de proporcionar comunicación de una forma rápida y efectiva. Los agentes autonomos tienen la capacidad inherente de comunicarse con otros agentes o usuarios por ello son ampliamente utilizados para desarrollar este tipo de aplicaciones.

Sensibilidad al entorno: con frecuencia los programas de aplicación necesitan tomar decisiones basadas en el estado de su entorno de operación, por ello las aplicaciones deben ser capaces de percibir su entorno y hacer una evaluación del mismo, para poder seleccionar el curso apropiado para la ejecución de una acción basada en los planes, objetivos e intenciones de esa aplicación, para luego ejecutar las acciones necesarias. Un software de agente inteligente proporciona los mecanismos para supervisar el entorno, dibujar inferencias, manipular símbolos y determinar el curso apropiado de una acción en una situación dada.

Se debe seleccionar una arquitectura de agente tal que cada uno de los agentes ejecute tareas específicas[83]. Estas redes o comunidades o redes de agentes se utilizan para comunicarse unos con otros y cooperar en la resolución de problemas.

83 http://www.wikilearning.com/implementacion_de_equipos_de_agentes_en_entornos_dinamicos_multi_agentes_i-wkccp-5095-23.htm

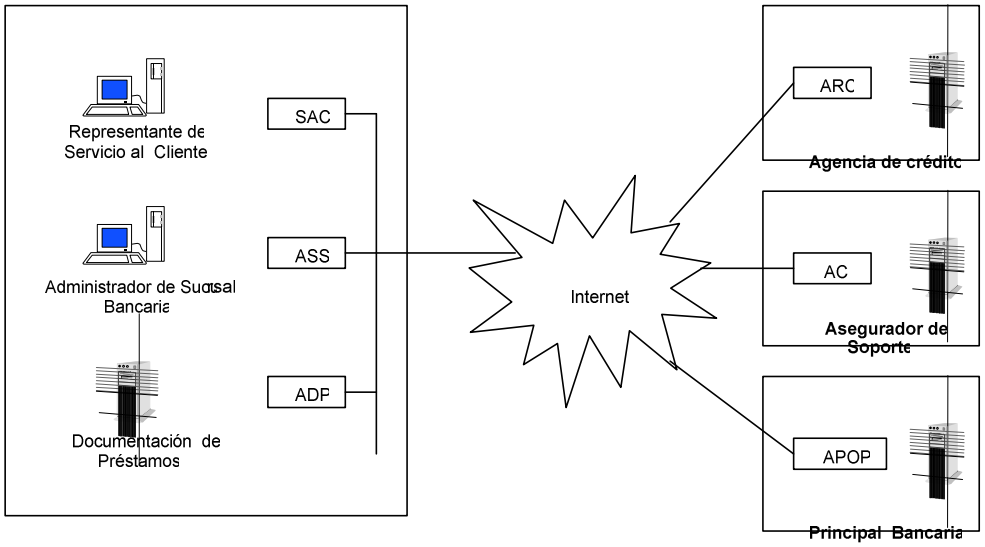
Los agentes se utilizan en una variedad de aplicaciones de negocios, manufactura, finanzas, viajes, industria de comercio electrónico, escritorios de ayuda, para implementar sistemas para búsqueda de información en Internet y otras muchas más.

4.1. Aplicaciónes bancarias

En una aplicación bancaria se pueden utilizar agentes para que desempeñen roles relacionados con el proceso de préstamo de dinero. El dominio de la aplicación consta de una sucursal bancaria, la oficina principal del banco, una agencia de préstamos e informe de créditos. Los participantes pueden estar geográficamente dispersos y se comunican a través de una red de datos por ejemplo Internet. El cliente generalmente trata con la sucursal del banco que es donde se inicia el proceso de aprobación del préstamo. El cliente trabaja con el representante de servicio al cliente quien reúne toda la información necesaria para otorgar el préstamo y crea una aplicación de préstamo de dinero, esta información se le envía al administrador de la sucursal bancaria para una revisión preliminar. El administrador luego contacta con al agencia de créditos para obtener el informe del crédito en un posible prestatario.

El informe del crédito se envía a la sucursal del banco donde el administrador de la sucursal revisa el informe y la aplicación emite una aprobación preliminar y envía el resultado al administrador de la oficina principal del banco. Si el crédito es aprobado, después de revisar el respaldo que ofrece el prestatario, entonces se envía finalmente al administrador de la oficina principal para que de la aprobación definitiva y la envíe a la sucursal para que el representante de servicio al cliente contacte con el usuario.

Figura 16. Software de agentes para procesos de préstamos bancarios



Todos los participantes en este escenario se pueden beneficiar del uso de la tecnología de agentes, por ejemplo una interfaz de agente inteligente en un Servicio de Atención al Cliente (SAC) puede ayudar al representante de servicio al cliente a recolectar la información necesaria y luego transmitirla al Agente Administrador de la Sucursal (AAS), el agente administrador puede revisar la aplicación del préstamo utilizando las mismas reglas de proceso que utiliza un administrador humano.

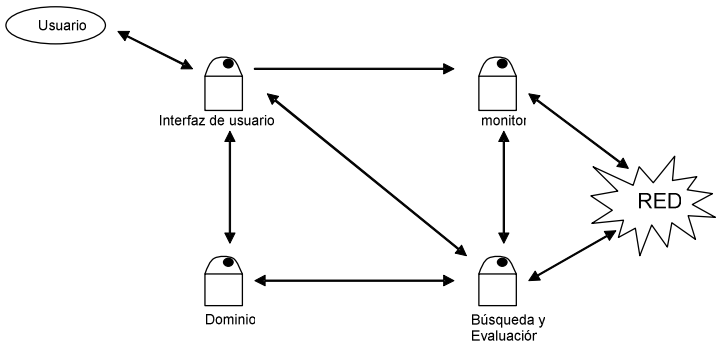
Un Agente de Prestamos de la Oficina Principal (APOP) puede comunicarse y compartir la información con los administradores de las sucursales. Algunos de los agentes involucrados en el proceso no pertenecen al banco, por ejemplo agencia de informe puede utilizar un Agente de Informe de Crédito (ARC) para investigar el historial de crédito a petición de un agente remoto. Se puede utilizar también un Agente Controlador de Información (ACI), para que controle la transferencia de la información de préstamo.

En la actualidad la información requerida se envía automáticamente y una Agente de Documentación de Préstamos (ADP) archiva toda la información tanto de la oficina principal como de la sucursal, además cada uno de los agentes puede tomar sus propias decisiones utilizando sus propias reglas para la aplicación de préstamos. El flujo de papeles dentro de la organización se reduce sustancialmente y se agilizan los procesos de aprobación.

4.2. Aplicaciones de escritorio

Con agentes es posible construir aplicaciones de escritorio de alto desempeño donde se utilizan los agentes inteligentes para desempeñar varias funciones. La siguiente figura representa la arquitectura para un asistente de escritorio inteligente que ayuda al usuario a encontrar y utilizar información desde Internet.

Figura 17. Uso de agentes en una aplicación de escritorio



Los cuatro agentes de la figura actúan por separado, cada uno de ellos especializado en una tarea particular, usados para crear una aplicación inteligente. Una agente de interfaz que maneja toda la interacción con el usuario, un agente monitor que supervisa los sitios Web de interés para el usuario y luego le informa (a través de la interfaz de agente) cuando parece información nueva en aquellos sitios.

Un agente de dominio es un agente inteligente que conoce las áreas de interés del usuario y finalmente el agente de evaluación y búsqueda que está especializado en localizar y evaluar la información en los sitios Web remotos y determinar si esa información satisface las necesidades del usuario. Los agentes monitor y de búsqueda/información son ejemplo de agentes de información.

Es claro entonces que al utilizar las facilidades de abstracción de los agentes inteligentes se puede hacer un buen diseño de un sistema. Cada agente se diseña para que ejecute una función particular, además cada uno se comunica con los demás, con el usuario y con la red mediante el uso de mensajes bien definidos. En este ejemplo todos los agentes se ejecutan sobre la misma plataforma y no necesitan comunicarse con cada uno de los demás sobre una red diferente. Utilizando agentes como una abstracción software de alto nivel se simplifican significativamente los procesos de desarrollo software.

Otros usos personales de los agentes: Pueden ayudar a que el usuario este informado de las cosas de casa o familia por ejemplo en lo referente a médico, automóvil, temas relacionados con la educación de los hijos, ofertas de empleo, vestimenta, servicio telefónico. Le recordará al usuario cuando tiene cita con el dentista, cuanto dinero debe y a quien, incluso pagará nuestras deudas con la tarjeta de crédito. En este sentido se puede convertir en el organizador de actividades, confidente financiero y hasta amigo a la hora de ayudarle al usuario a resolver problemas personales.

4.3. Integración a la empresa

Los sistemas clásicos de información no han sido diseñados para manejar la variabilidad y flexibilidad requerida por las empresas modernas, los sistemas de información tienen que moverse desde un ordenador central orientado al entorno, donde todos los datos y el poder de computo se centra en un lugar a aplicaciones cliente/servidor donde el poder de computación se propaga a través de la red, pero los datos están centralizados.

El paso siguiente en un sistema de cómputo es lograr que tanto los datos como el poder de computación se distribuyan a través de la red formando lo que se conoce como entornos de computación distribuidos. Como integrantes de la empresa los agentes tienen la habilidad de unir sistemas de información diversa e incompatible a través de la red o de Internet, pueden almacenar datos y presentarlos en una variedad de formas convirtiéndolo al formato requerido por el receptor sobre la marcha. La tecnología de agentes permite el desarrollo de aplicaciones de forma rápida y con soporte para una gran variedad de sistemas.

- Filtrado y enrutamiento de la información: Muchos de los sistemas de información confunden los datos con información propiamente dicha, esta no es filtrada ni personalizada. Para incrementar la productividad el sistema necesita manejar la información por excepciones y no por volumen, los agentes inteligentes dan la información necesaria que se desea conocer en el momento justo en el que se necesita ya que el agente actúa aplicando las reglas de negocio y de esa forma libera al ejecutivo de tareas ahorrándole tiempo y mejorando la comunicación.

- En la mira de los nuevos negocios está proporcionar las bases para una nueva generación los Sistemas de Información Ejecutivos.
- Seguridad: Con la integración de Internet y el comercio electrónico los agentes actúan como policías o como salta fuegos para proteger las bases de datos de la empresa, pero esta habilidad exige sistemas más robustos para responder a las necesidades de la empresa. Los agentes inteligentes actualmente mantienen el control de acceso y eliminan las múltiples claves de acceso mejorando la productividad ejecutiva y la seguridad, además pueden cambiar las claves de acceso al sistema diariamente o por horas según lo requiera el propio sistema y mueven los datos sobre redes realmente seguras y fiables pues lo que les interesa es que la información llegue segura a su destino.
- Apoyo: Los agentes inteligentes pueden actuar como apoyo para los ejecutivos ejecutando tareas por ellos o soportándolos en conocimientos que ellos no poseen. Muchas de las tareas en el campo de los negocios siguen reglas estándar, la habilidad de los agentes inteligentes para desempeñar esas tareas siguiendo las reglas y actuar independientemente, son factores claves que diferencian el software tradicional del software de agentes. La productividad de una empresa puede cambiar dramáticamente cuando se incorpora los agentes inteligentes para dar apoyo y esto lo pueden hacer de varias formas:

    - Dando orden de aprobación automática si el cliente se encuentra en las condiciones financieras necesarias.
    - Solicitando recursos cuando al hacer el inventario se encuentra que las existencias son bajas.

- Cambiando las reservas de pasaje cuando los vuelos son cancelados.
- Realizando compra de productos al mejor precio vía Internet.
- Actuando como supervisor de los sistemas menos inteligentes.

Puede resultar difícil que una sola persona realice todas estas tareas, pero si cuenta con el apoyo de un agente el trabajo se realizará más rápidamente y con menos errores, en este sentido los agentes pueden actuar como apoyo para departamentos individuales.

4.4. Informática basada en agentes

Los beneficios de utilizar una arquitectura informática basada en agentes son:

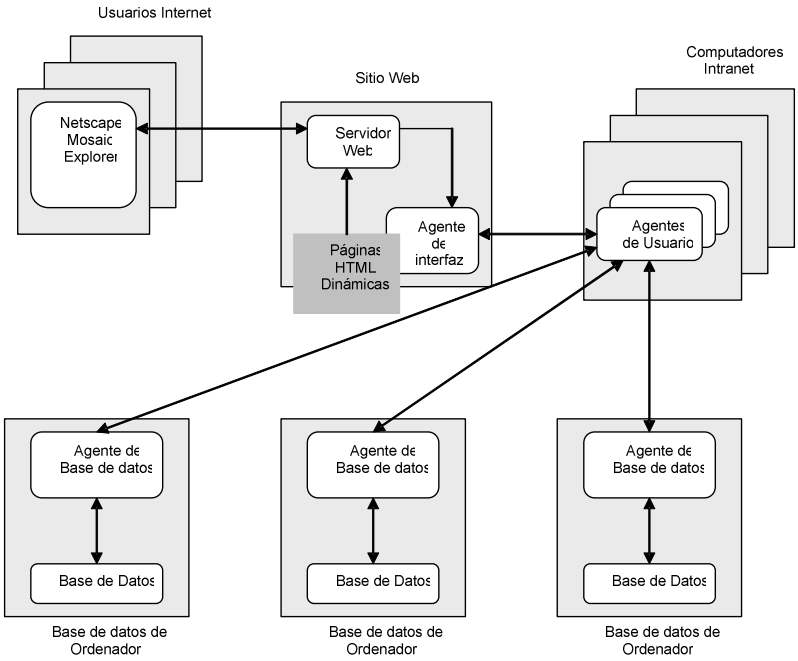
- Desarrollo rápido: los agentes pueden desarrollar varias tareas en paralelo
- Robustez y seguridad: el sistema puede mantener la operación y reducir la funcionalidad si algunos procesos y la red caen y recobrar toda la funcionalidad una vez los procesos operen de nuevo.
- Rápida respuesta a los cambios de requisitos: el sistema puede fácilmente cambiar las reglas existentes dentro de los agentes para se que acoplen a los nuevos requisitos.
- Desempeño: La ejecución de las tareas en diferentes ordenadores
- Seguridad: El acceso de datos es controlado por los agentes de acuerdo a los privilegios de acceso, también la arquitectura de agentes es compatible con los estándares de seguridad de Internet.

La arquitectura de una aplicación Web basada en agentes utiliza un buscador convencional como Netscape, la página del usuario ubicada en el sitio Web es una página estática, esta página de entrada contiene un script CGI que hace que el servidor Web pase la respuesta a un agente de interfaz el cual pasa una página HTML generada dinámicamente y la envía al servidor Web el cual pasa la página al buscador del usuario como si hubiese sido generado estáticamente.

Estas páginas generadas dinámicamente tienen empotrados scripts CGI que hacen que su respuesta pase al agente de interfaz. Cuando el usuario pasa la página se crea un agente de usuario para que maneje todo el diálogo con el usuario, estos agentes se basan en un prototipo común para todos los agentes de usuario. Los agentes de usuario pueden ejecutarse en el sitio Web por ellos mismos, en sistemas más complejos los agentes se pueden ejecutar en múltiples ordenadores para hacer que el sistema maneje varios usuarios al tiempo.

Los agentes de usuario son responsables de generar dinámicamente páginas HTML en respuesta a los requisitos del usuario. Para hacer esto solicitan información de las diferentes bases de datos enviando mensajes a los agentes de bases de datos que controlan el acceso a las bases de datos. Los agentes de usuario pueden también enviar información que reciben de los usuarios a los agentes de bases de datos para entrar en las bases de datos. La ventaja de esta arquitectura es que el formato estándar de los mensajes se puede intercambiar entre los agentes de usuario y los agentes de bases de datos los cuales contienen todo el conocimiento del formato específico de las bases de datos y como responder a la alta demanda de peticiones de información.

Figura 18. Aplicación web basada en agentes



Esto facilita la incorporación de nuevas bases de datos al sistema sin necesidad de cambiar los agentes de usuario el cual realiza una gran cantidad de intercambio de datos con los agentes de bases de datos. Esta arquitectura de agentes es inherentemente asíncrona y el acceso a las bases de datos se hace en respuesta a las peticiones de múltiples usuarios.Un agente de base de datos puede hacer muchos accesos a disco para generar una respuesta a la petición de alto nivel de un usuario. Esto simplifica el tráfico en la red ya que los accesos se hacen localmente. Por otra parte también actúan como barrera para restringir el acceso de datos sensitivos. El agente de usuario se encarga de solicitar la clave de acceso y envía esa información al agente verificador quien se encarga de autentificar los accesos y mirar los privilegios para el usuario y envía esa información al agente de usuario quien la utiliza para controlar la información presentada a cada usuario[84].

84 http://cita2003.fing.edu.uy/articulosvf/42.pdf

4.5. Aplicaciones industriales

La tecnología de agentes es aplicada en sectores industriales.

4.5.1. Procesos de control

Esta es una aplicación natural de agentes inteligentes y sistemas multiagente[85], los controladores de procesos son sistemas reactivos autónomos el más conocido de ellos es ARCHON que es una plataforma software para construir sistemas multiagente, se ha aplicado a varias aplicaciones de control de procesos incluyendo el manejo de transporte de electricidad y el control de un acelerador de partículas. Los agentes son sistemas computacionales con cuatro componentes principales, un módulo de comunicación de alto nivel (HLCM) que se encarga de manejar la comunicación entre agentes, un módulo de planeación y coordinación (PCM) que decide lo que el agente hará, un módulo de agente de gestión de información (AIM) responsable de mantener el modelo de agentes de le mundo y finalmente un sistema inteligente básico (IS) que representa el dominio de experiencia del agente que se puede utilizar para encapsular un sistema inteligente existente y enviarlo a un agente.

4.5.2. Procesos de fabricación

Un sistema de manufactura conocido es YAMS (Yet Another Manufacturing System) que aplica el Protocolo de Contrato de Red para el control de fabricación. El objetivo de este sistema es el manejo eficiente de los procesos de producción los cuales se definen como parámetros cambiantes tales como los productos a ser fabricados, recursos disponibles, limitaciones de tiempo entre otros.

---

85 Victor Carneiro, Vicente Orjales, Justo Hidalgo, "Gestión de Calidad de Servicio en Aplicaciones Web",http://www.upv.es/sma/teoria/aplicaciones/industria.pdf.

Para realizar estas tareas complejas YAMS adopta una aproximación multiagente donde cada fábrica y componente de la fábrica se representa como un agente. El protocolo de contrato de red permite que las tareas sean delegadas a factorías individuales de allí va a los Sistemas de Manufactura Flexible (FMS) y luego a las celdas individuales.

4.5.3. Control de tráfico aéreo

Uno de los sistemas utilizados para control de tráfico aéreo es el OASIS. La metáfora del agente proporciona una forma útil y natural de modelar los componentes autónomos del mundo real, el sistema permite que el agente se implemente utilizando el modelo de intención - deseo - creencia que es una de las aproximaciones más populares de razonamiento en sistemas multiagente teóricos.

4.6. Aplicaciones comerciales

4.6.1. Gestión de la información

La riqueza y variedad de información disponible hace necesario contar con herramientas que permitan manejar la información y tratar de evitar el problema de sobre carga en la red. Los factores humanos y los organizativos atentan contra el uso de los recursos de una forma sistemática. Se puede caracterizar los problemas de sobrecarga de información de dos formas:

Filtrado de información: Cada día nos encontramos con gran cantidad de información (por ejemplo la recibida vía correo electrónico o de las news) y solamente una pequeña porción de la misma es relevante, entonces es necesario que nos centremos en lo nos interesa realmente.

Concurrencia de la información: El volumen de información disponible impide que encontremos información a una respuesta específica. Debemos ser capaces de encontrar la información que se ajusta a nuestros requisitos. Hay muchos proyectos orientados a hacer este tipo de trabajo:

- Maxims: Describe un agente de filtrado para correo electrónico, el programa aprende a priorizar, borrar, devolver, cortar, archivar mensajes de correo. Constantemente realiza predicciones internas para saber lo que el usuario hará con los mensajes de correo, en otros casos le sugiere al usuario que hacer con el correo.
- Newt: Este es un programa que filtra noticias de Internet, está implementado en C++ sobre una plataforma Unix, toma una serie de artículos y de a cuerdo a las preferencias del usuario selecciona los que considera recomendables para que el usuario lea. El agente Newt se programa por medio de ejemplos de entrenamiento, el usuario le da artículos que debe y no debe leer para que aprenda o puede también programar al agente con reglas explícitas (por ejemplo "dame todos los artículos que contengan la palabra agente") guarda las palabras y luego realiza un análisis textual de los documentos para encontrar las palabras clave.
- The Zuno Digital Library: una librería es una organización que maneja una colección de datos. El ZDL es un sistema multiagente que permite que el usuario tenga una vista organizada de una fuente de colección de datos desorganizados tal como WWW. Los agentes juegan tres roles principales:
    - Consumidor: representan al usuario final del sistema.
    - Productor: Representa un proveedor de contenido que se apropia de la información que el consumidor toma.
    - Facilitador: Un puente entre consumidor y proveedor.

Los agentes consumidores son responsables de representar los intereses del usuario, mantienen los modelos del usuario y los utilizan para asistirlo proporcionándole la información que le interesa. El sistema ZDL actúa como un filtro de información y recolector de información.

La tecnología de los buscadores al menos como funciona hoy en día no soluciona el problema de que de tanta información podamos tener la que realmente nos interesa, es claro entonces que una posible solución sea la utilización de agentes ya que son herramientas ideales para buscar, filtrar, organizar información. En cuanto a las empresas los pueden utilizar para personalizar la información que ofrece a sus clientes y optimizar el acceso a la información que ofrece en sus Intranets. A los usuarios nos ayudan en nuestra navegación y búsqueda de información en Internet.

Existen muchas empresas que utilizan la tecnología de agentes para ofrecer información personalizada a sus usuarios. Por ejemplo la empresa Time Inc. New Media ofrece su popular servicio Pathfinder que es una versión electrónica de las revistas Money, People, Sport Illustred, Fortune, entre otras, lo interesante es que ofrece un servicio denominado Personal Edition que consiste en un sistema personalizado de noticias, el suscriptor puede construir una serie de consultas que se usarán para crear una edición personalizada en Web de las noticias. Otro servicio parecido es el que ofrece el periódico de San José Mercury News que es uno de los pioneros en este tema. Con NewsHound los usuarios pueden crear hasta cinco perfiles especificando los términos obligados, opcionales y excluyentes que describen el tema de interés.

A medida que las noticias interesantes llegan al servicio estas se envían por correo electrónico cada hora a lo largo de todo el día.

Internet se ah convertido en un poderoso medio para gestionar y compartir información dentro de una empresa y el uso de agentes como se ha mencionado antes optimiza el uso y acceso a esta. Con agentes que trabajen en el servidor y que permitan ser configurados desde el navegador del usuario sin que tenga la necesidad de mirar constantemente si se han realizado cambios en la información corporativa, puesto que esa información llega directamente al ordenador del usuario o al menos la notificación de que se han producido.

Para los usuarios interesados en buscar información existe por ejemplo un programa de IBM el WBI (Web Browser Intelligent) que proporciona los siguientes servicios[86]:

- Personal History: recuerda los sitios visitados, recoge la información sobre ellos (título, contenido), para realizar luego búsquedas en función de las palabras clave.
- Internet Watch: Vigila si se han realizado cambios en los sitios que al usuario le interesan y le avisa de los cambios ocurrido.
- Offline Browsing: Se encarga de bajar páginas de interés. Se puede configurar para que trabaje en las horas de menos tráfico o en los momentos en los que el usuario no esté trabajando con Internet. Una vez que las páginas están en el disco duro las organiza convenientemente.
- Path: Ayuda a encontrar páginas en las que habías estado pero no recordabas como volver.

86 "Gestión de Calidad de Servicio en Aplicaciones Web", http://cita2003.fing.edu.uy/articulosvf/42.pdf.

- Shortcuts: Este servicio ayuda a navegar de forma más eficiente de forma más eficiente por la red, el agente está atento a los hábitos de navegación y comportamiento del usuario con lo que para búsquedas futuras le proponga caminos alternativos más cortos para llegar a la página que le interesa eliminando el paso por páginas intermedias y envía mensajes del tipo: " he notado que desde aquí vas a alguna de las siguientes páginas, selecciona una de ellas si quieres ir allí".

Existen otros programas que realizan tareas parecidas como WebCompas que busca y organiza la información y baja solo la información y no la página.

Otro de los servicios ofrecido por Internet son los grupos de news y resulta difícil filtrar tal cantidad de información, pero utilizando programas del tipo de News Monger se puede realizar búsquedas de dos formas: una directamente sobre el buscador de la base de datos de Altavista y otra directamente sobre los grupos de noticias que nos interesen[87].

4.6.2. Comercio electrónico

El comercio electrónico es una realidad en Internet y cada vez es mayor el número de personas que realizan sus compras a través de la red. En este campo los agentes pueden prestar su servicio no solo al comprador sino al vendedor.

87 "Secretos de los multibuscadores, los buscadores de personas y el software especializado para encontrar páginas y archivos en Internet", http://www.idg.es/iworld/articulo.asp?id=28917.

Al vendedor le ayudan a contactar con el cliente, a captar nuevos clientes y a crear sus perfiles para poder ofrecer un servicio más personalizado. Por ejemplo en la página de ComputerESP a través de una gente de puede obtener información puntual sobre componentes específicos HW o SW de varias compañías proveedoras. El agente aprende y recuerda los productos que el cliente ha estado consultando y utiliza esa información para orientar a los clientes novatos. Otro servicio que ofrecen es el de encontrar el producto que queremos al mejor precio en las diferentes tiendas virtuales existentes en Internet. Por ejemplo:

- BargainFinder que es un agente que se puede configurar para que busque el artículo con un precio de referencia. Este agente ha sido diseñado por Andersen Consulting y funciona desde 1995 y se está utilizando para estudiar los hábitos de compra de los usuarios de Internet.
- Jango es otro producto que permite escoger entre once categorías de productos por ejemplo si se quiere comprar un libro pide el título o el autor, en caso de un vino pide el nombre de la bodega productora o el tipo de vino. Luego de buscar por diferentes cibertiendas envía el detalle de la información encontrada organizada por precios.

4.6.3. Gestión de procesos comerciales

Los directores de diferentes compañías realizan informes de las decisiones tomadas basándose en la combinación de información y juicios recibidos de los diferentes departamentos. Obtener información pertinente y actualizada en una compañía muy grande puede ser una tarea bastante engorrosa y que consuma mucho tiempo.

Por esta razón algunas empresas han desarrollado sistemas que actúen como asistentes en varias de los aspectos de gestión de procesos comerciales. Por ejemplo el proyecto ADEPT ataca el problema viendo los procesos como una comunidad de agentes negociantes y proveedores de servicio. Así cada agente representa a un departamento diferente con un rol distinto dentro de la empresa y que es capaz de proporcionar más de un servicio.

Por ejemplo el departamento de diseño puede proporcionar un diseño de una red de telecomunicación, el departamento de logística se encarga de verificar que cumpla con las condiciones legales necesarias para su funcionamiento, el departamento de marketing puede proporcionar el costo del diseño, etc. También se requiere el servicio de otro agente que pueda entrar en negociaciones para que el servicio obtenido sea aceptable en precio, tiempo y grado de calidad. Esta aproximación basada en agentes ofrece muchas ventajas sobre la típica solución de Workflow para este tipo e problemas. La naturaleza proactiva de los agentes permite que las excepciones en el servicio puedan ser detectadas y manejadas de una manera flexible.

4.7. Aplicaciones médicas

La informática médica es un área dentro de las ciencias de computación que está creciendo se encuentran nuevas aplicaciones para computadores en la industria de la asistencia sanitaria.[88]

4.7.1. Supervisión de pacientes

El sistema guardián intenta ayudar a supervisar el cuidado de los pacientes en una unidad quirúrgica de cuidados intensivos.

88 “Tecnologías de la información al servicio de la historia clínica electrónica”, ww.conganat.org/Seis/informes/2003/PDF/CAPITULO6.pdf.

Hay dos motivos principales que motivaron la creación de este sistema, primero que para el cuidado del paciente es esencial que el grupo de personas que lo tienen a su cargo cooperen en la organización del cuidado del paciente y segundo que compartan la información concerniente al estado del paciente. En particular los especialistas tienen muy poco tiempo para supervisar minuto a minuto el estado de sus pacientes, esta tarea se deja por completo a las enfermeras quienes no tienen la experiencia para interpretar la información que obtienen en la forma en que el experto lo haría. El sistema guardián distribuye la función de supervisión del paciente en la UQCI entre un número de agentes de tres tipos diferentes:

- Agentes de Percepción/Acción: responsables de la interfaz entre el sistema Guardián y el mundo.
- Agentes Racionales: Organizan los procesos de toma de decisión del sistema.
- Agentes de Control: Este agente es único y se encuentra en la parte más alta de control de la jerarquía del sistema.

Los agentes tienen una organización jerárquica basada en un modelo de control en el que diferentes agentes cooperan a través de la contribución al conocimiento en una base de datos estructurada común.

4.7.2. Asistencia Sanitaria

Existe un prototipo de un sistema distribuido de asistencia sanitaria. Este sistema está diseñado para integrar los procesos de manejo del paciente. Por ejemplo un médico general puede sospechar que un paciente tiene cáncer peno no puede confirmar su sospecha sin la asistencia de un especialista, una vez confirmado el programa de asistencia debe crear un tratamiento para el paciente.

El sistema prototipo permite una presentación natural de estos procesos asignando un agente a cada individuo y potencialmente a cada organización relacionada con los procesos de asistencia sanitaria de los pacientes. Los agentes en el prototipo contienen un sistema inteligente de conocimientos que contienen el dominio de experiencia del agente, una interfaz hombre máquina, que permite al usuario adicionar, remover o mirar los objetivos del sistema y un administrador de las comunicaciones.

4.8. Entretenimiento

La industria del tiempo libre es con frecuencia tomada con muy poca seriedad por la comunidad de las ciencias de computación. Las aplicaciones en este campo se miran como algo periférico, sin embargo las aplicaciones como juegos de ordenador pueden ser extremadamente lucrativas. Los agentes tienen un rol en los juegos de computador, en el teatro interactivo y en las aplicaciones de realidad virtual. Tales sistemas tienden a ser caracteres animados autónomos o semi autónomos que pueden implementarse como agentes.

4.8.1. Juegos

Dentro de la tecnología de agentes aplicada a los juegos de computador se encuentra un ejemplo del popular juego de computador Tetris en el que el usuario debe construir un muro a partir de bloques de forma irregular que van cayendo. El agente hace parte del usuario quien va controlando donde van cayendo los bloques.

Tratar de programar este tipo de agentes utilizando las técnicas simbólicas tradicionales de IA requerirá ir a través de diferentes estados de conocimiento, representando le conocimiento del juego y el rol del usuario en términos de estructuras de datos simbólicas tales como reglas entre otras. Esta aproximación puede resultar ser totalmente irreal para un juego como Tetris el cual tiene fuertes restricciones de tiempo real.

Se utiliza entonces un modelo de agente reactivo como alternativa llamado RTA (Real Time Able). En esta aproximación los agentes se programan en términos de comportamientos que son estructuras simples que se asemejan a reglas pero que no requieren razonamientos simbólicos complejos.

4.9. Educación

La tecnología de agentes se utiliza en la educación por ejemplo, para dar soporte a actividades de colaboración en entornos de aprendizaje cooperativo asistido por computador (CSCL)[89]. Otro tipo de sistemas que incorpora la tecnología de agentes son los Sistemas Hipermedia Adaptativos en los que las funciones inteligentes del sistema están a cargo de los agentes que actúan como tutores en el aprendizaje para lo cual generan un perfil del alumno basado en sus conocimientos con lo cual determinan las áreas del curso que le conviene visitar y además lleva un registro del progreso del alumno en las diferentes áreas estudiadas.

89 Guillermina Waldegg Casanova, "El uso de las nuevas tecnologías para la enseñanza y el aprendizaje de las ciencias", http://redie.uabc.mx/vol4no1/contenido-waldegg.pdf.

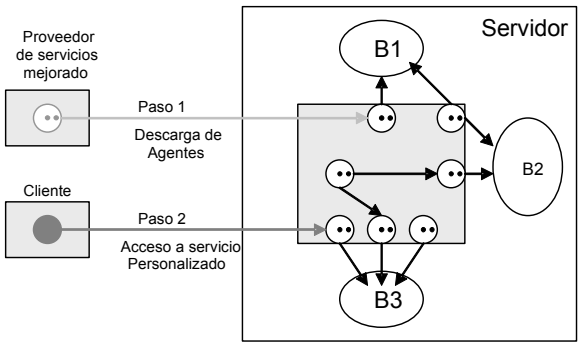
5. IMPACTO DE LOS AGENTES SOBRE LAS ARQUITECTURAS DE SERVICIO

El comercio electrónico es una muestra de cómo la tecnología de agentes permite realizar nuevas arquitecturas de servicio. Un prerequisito para hacer de un comercio electrónico un lugar vital y dinámico como un comercio real es que potencialmente todo el mundo pueda proporcionar y explotar servicios. En un comercio electrónico la tecnología de agentes puede resaltar arquitecturas para facilitar la caracterización y provisión instantánea de servicios.

5.1. Caracterización y configuración de servicios

La posibilidad de personalización y configuración de servicios proporciona las bases para un correo altamente dinámico. En general no todas los servicios pueden implementarse utilizando tecnologías de Inteligencia Artificial, consecuentemente, el comercio electrónico debe basarse sobre servicios que han sido y serán realizados utilizando tecnología convencional, esos servicios se conocen como servicios básicos. Gracias a la movilidad de los agentes se puede personalizar y los nuevos servicios se pueden establecer configurando los existentes.

Fig. 19. Personalización de servicios básicos



Para hacer esto el estilo de ejecución remota de movilidad es suficiente, en otras palabras los agentes se pueden mover a un nodo y permanecer hay hasta que terminen. Los servicios se pueden personalizar o mejorar de varias formas, enviando agentes a las correspondientes máquinas servidoras[90], la anterior figura ilustra un escenario con tres servidores (B1, B2, B3) y un número de agentes que personalizan esos servicios. Los agentes operan en un dominio protegido de la máquina servidora y proporcionan una interfaz de servicio personalizado a otros agentes y acceso a servicios locales básicos sobre la máquina servidora. En otras palabras este tipo de agentes se puede ver como servicios básicos terminales.

Cada servicio básico puede ser personalizado o extendido de varias formas, por ejemplo, el servidor B3 tiene tres terminales con los cuales puede implementar servicios completamente diferentes. En el caso de comercio electrónico es concebible tener varios agentes compitiendo y proporcionando servicios idénticos o similares a sus clientes.

Para proporcionar un servicio personalizado un cliente simplemente tiene que desarrollar el agente apropiado y transferirlo a la máquina servidora correspondiente. El servicio es instaurado en el momento que llega a la máquina servidora y no se requiere de más funciones, en este caso el agente proporciona un nuevo servicio usando uno o más servicios existentes. Por supuesto, los nuevos servicios se pueden configurar dinámicamente accediendo a múltiples servicios básicos o a servicios personalizados.

Se puede esperar que haya diferentes niveles de servicio, en particular el rol de vendedores de servicios está emergiendo en este contexto.

---

90 http://software.elpais.com/rss/1015/last_news_by_title.xml

5.2. Utilización del servicio instantáneo

Para poder acceder a un servicio remoto un usuario final tiene que ejecutar un programa cliente local con la cual implementa una interfaz de usuario y contiene además la lógica para el acceso al servicio remoto. Con la tecnología convencional el usuario puede tener el programa cliente apropiado por ejemplo desde un lector de CD e instalarlo en su máquina antes de usar el servicio. La tecnología de agentes permite crear nuevos clientes de servicio. Para interactuar con un cliente potencial, el proveedor de servicios debe crear un cliente de servicio apropiado con el cual viaje hasta el cliente. Al recibir un servicio de un cliente de servicio el usuario final puede inmediatamente utilizarlo. La diferencia de bajar código ejecutable es que los agentes se escriben en lenguajes de programación seguros generalmente propietarios en los sistemas abiertos, y que están listos para ser ejecutados.

La bajada de agentes se puede hacer por peticiones del usuario final o por parte del proveedor de servicios. En el primer caso el usuario final que desea utilizar el servicio puede recibir el cliente de servicio móvil correspondiente desde un directorio o desde un negociador de servicios. En lugar de regresar a un identificador el directorio puede crear un cliente de servicio móvil que viaje hasta el usuario. En el segundo caso el proveedor de servicios puede activamente enviar un nuevo cliente de servicio a un cliente potencial ejecutando alguna clase de negociación activa. Esta clase de clientes de servicio móviles pueden no solamente informar al usuario acerca de los nuevos servicios sino también ofrecer una prueba de tales servicios, con esto el concepto de negociación activa da una nueva dimensión a la informática cliente/servidor. Se pueden distinguir tres clases diferentes de clientes de servicio según su complejidad:

5.2.1. Clase de agentes simples

Esta clase de agentes actúan como agentes de servicio o como mensajeros. De acuerdo a la distinción entre ejecución remota y migración, existen dos escenarios alternativos para el procesamiento de servicios: Adoptar un escenario de ejecución remota: el agente móvil realiza el servicio completo por ejemplo contiene toda la lógica del servicio y datos. Una vez que llega a la máquina cliente el agente de servicio comienza a dialogar con el usuario y este le hace la petición del servicio que desea, proporcionando un registro de información de forma animada. Una vez terminado la fase de procesamiento de servicio el agente se auto destruye.

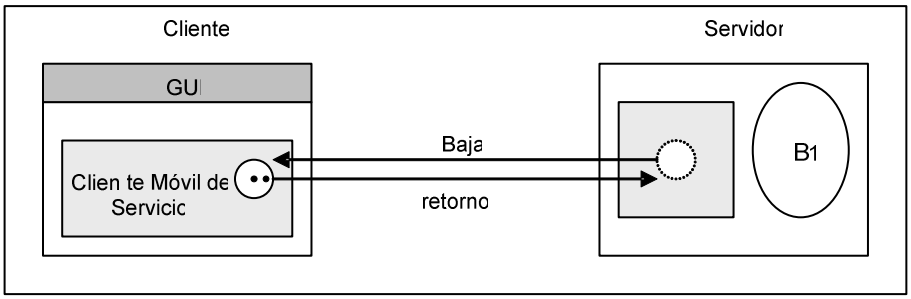
En el caso de un escenario de migración el agente móvil de servicio es capaz de regresar al servicio básico antes de interactuar con el cliente. Esta clase de agentes móviles de servicio se pueden implementar a partir del estado del arte de la tecnología de agentes. Mientras este tipo de agentes móviles de servicio es apropiado para servicios simples (como los servicios de petición), también son simples para muchos servicios complejos.

5.2.2. Clase de agentes cliente/servidor

Los agentes de esta clase representan los programas de clientes móviles para entornos cliente/servidor. Al igual que la clase de agentes simples los agentes de esta clase implementan interfaces de usuario y dialogan con el usuario. Además pueden interactuar directamente con servicios básicos remotos de acuerdo con un protocolo apropiado cliente/servidor.

La siguiente figura ilustra como se realiza la comunicación entre cliente móvil de servicio y el servidor. Si el cliente de servicio móvil lleva una definición abstracta de un servicio, tal como una especificación IDL de una interfaz de servicio, se puede utilizar un sistema estándar RPC o un servicio similar. Un candidato atractivo es CORBA de OMG, ya que esta tecnología promete estandarizar el acceso a un amplio rango de servicios y sistemas.

Figura 20. Un cliente móvil de servicio cliente/servidor



5.2.3. Clase de agentes multimedia

La clase multimedia también utiliza el concepto de movilidad del cliente de servicio, sin embargo esta clase de agentes le da al usuario acceso a datos multimedia incluyendo medios continuos como audio y vídeo, por ejemplo un cliente móvil de servicio puede ofrecer muestras de servicios de vídeo sobre demanda ofreciendo películas y juegos interactivos. Los objetos multimedia pueden estar formados por múltiples objetos multimedia y objetos monomedia, entre los que se puede definir varios relaciones temporales y especiales. Un objeto multimedia se dice que es interactivo si permite la interacción del usuario. La integración de la tecnología de agentes y la multimedia conduce al usuario hacia clientes móviles de servicio multimedia.

Tales agentes no solamente proporcionan un acceso inmediato a servicios multimedia complejos, sino que también, sino que también explotan la tecnología multimedia interactiva para realizar interfaces de usuario sofisticadas.

Obviamente los agentes de la clase multimedia son más difíciles de construir, sin embargo hay una tecnología emergente que se basa en el uso del estándar MHEG que se encarga de problemas relacionados con objetos multimedia complejos. El lenguaje MHEG se adapta fácilmente al lenguaje de agente, de allí que ambos sean fácilmente interpretados. MHEG-3 define un lenguaje script para objetos MHEG, la figura anterior muestra un escenario con MHEG basado en un cliente móvil de servicio. El entorno de computación de un agente en un sistema de usuario final debe proporcionar agentes estacionarios específicos o herramientas para acceder a los llamados motores MHEG que son los responsables de la orquestación o coordinación de la presentación de objetos multimedia. El sistema de transporte en tiempo real se utiliza para acceder a la información multimedia por ejemplo al contenido de los objetos almacenados en diferentes servidores.

5.3. Impacto de los agentes sobre las telecomunicaciones

El área de las telecomunicaciones representa uno de los campos más cambiantes dentro de la emergente tecnología de agentes, en particular los campos de comunicaciones personales/inteligentes y la gestión están en las miras de investigación[91]. La inteligencia artificial proporciona una mejor distribución en el control y gestión de sistemas de telecomunicaciones. Generalmente los agentes móviles son capaces de colocar control y procesos software de gestión en los lugares más apropiados dentro de un entorno de telecomunicaciones. Esto tendrá un impacto significativo sobre las arquitecturas y los protocolos relacionados de sistemas de telecomunicación existentes.

Los entornos actuales de telecomunicación se basan en estándares IN (Intelligent Network) y TMN (Telecomunications Management Network), representan los fundamentos para la creación uniforme y eficiente, provisión y gestión de servicios de telecomunicación avanzados. Actualmente IN y TMN se basan en aproximaciones altamente centralizadas para la ubicación de servicios de control y gestión inteligente de redes donde los protocolos relacionados se basan en el paradigma tradicional cliente/servidor. Esto significa que los nodos centralizados conocidos como SCPs (Service Control Points), y OSs (Operation Systems), se utilizan para servicios de hospedaje y programas de gestión, respectivamente.

5.3.1. Telecomunicaciones basadas en Agentes

En general se puede identificar dos aproximaciones básicas sobre la arquitectura de servicios basados en agentes:

91 Tecnología y Telecomunicaciones, http://hiddekelmorrison.blogspot.com/

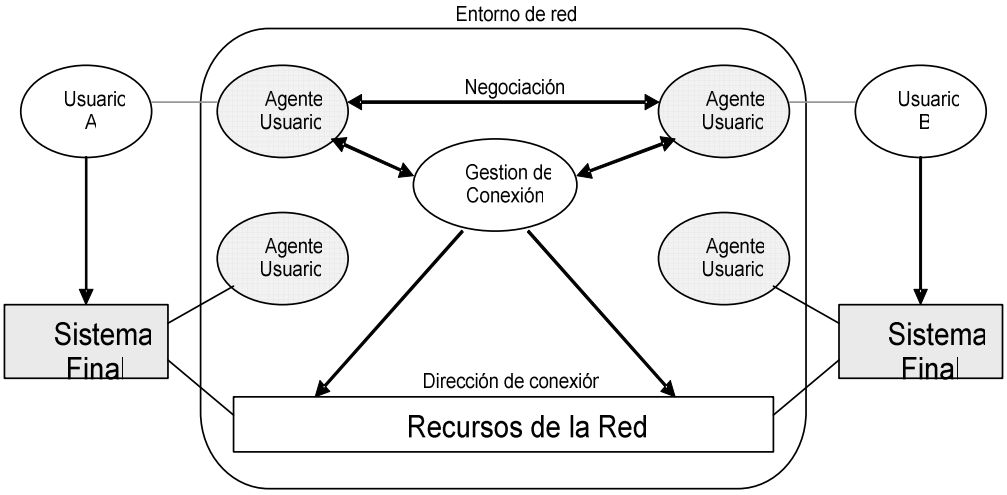
- Redes Listas: Los agentes son entidades estacionarias en la red, poseen la inteligencia necesaria para realizar autónomamente tareas específicas predeterminadas. Los atributos básicos de este tipo de agentes son su habilidad para actuar asíncronamente, comunicarse, cooperar con otros agentes y ser dinámicamente configurables. Esta clase de agentes estáticos tales como los agentes de usuario o los agentes de gestión se pueden considerar más como un entorno de ejecución de agentes el cual ejecuta scripts (por ejemplo, el tipo de agentes de ejecución remota).
- Mensajes Listos: Los agentes son entidades móviles que viajan entre diferentes sistemas/ordenadores y ejecutan tareas específicas en sitios remotos. Los agentes móviles contienen toda la información necesaria, en lugar de los nodos y sistemas finales correspondientes dentro de la red. Esto significa que los correspondientes entornos de ejecución de agentes deben ser proporcionados por los sistemas de usuarios finales potenciales y dentro de la red para desempañar la ejecución de agentes y la realización de los propuestos.

5.3.2 Comunicaciones Inteligentes basadas en Agentes

En el caso de un sistema listo el acercamiento a agentes estáticos será desarrollado en el sistema/red. Aquí un agente es un pequeño asistente personal, comúnmente referido como un agente de usuario o agente personal, que conoce las preferencias de comunicación de su usuario con respecto al tiempo, espacio, costo, medio, seguridad, calidad, accesibilidad y privacidad.

En nombre del usuario el controla toda la información que entra y sale con respecto al enrutamiento inteligente, el filtrado de información y servicio entre redes de trabajo (por ejemplo conversión de información). Los respectivos agentes de usuarios de los socios de comunicación involucrados por ejemplo, parte llamada y llamante (que pueden ser más de dos en el caso de llamadas multiparte), tienen que negociar y cooperar para establecer la sesión de comunicación deseada entre los usuarios. Esto se aplica en ambos casos de servicios asíncronos (mail) y síncronos (telefonía).

Figura 21. Telecomunicaciones basadas en agentes estáticos

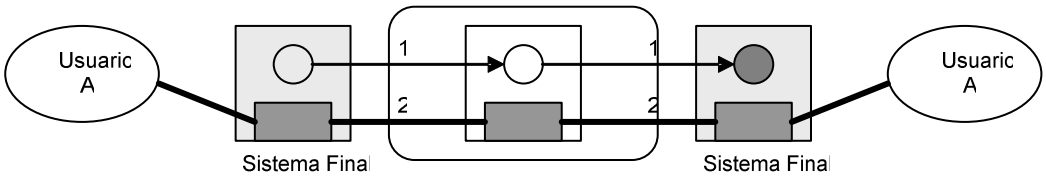


En particular, TINA se considera como una arquitectura IN y define una arquitectura similar[92]. En al actualidad, se esta realizando muchas actividades de investigación en el área de comunicaciones inteligentes/personales, las cuales introducen la noción de agentes de usuario.

92 Antonio Vallecillo Moreno, "RM-ODP: El Modelo de Referencia de ISO para el Procesamiento Abierto y Distribuido", http://congresos.lcc.uma.es/~av/Publicaciones/00/odpesp.pdf,

Adoptando la aproximación de mensaje listo los agentes móviles se pueden utilizar en dos dominios para intercambio asíncrono de información tales como servicios de correo y para pre-establecer intercambio de información en tiempo real, tales como los servicios de información multimedia. En el primer caso el agente transfiere la información deseada a través de la red (por ejemplo correo multimedia), en el segundo caso el agente móvil se puede utilizar principalmente para señalización y configuración del sistema por ejemplo establecimiento de un servicio de comunicación en tiempo real como vídeo conferencia). Esto significa que un agente puede ser generado por una llamada del sistema de usuario final previa a una sesión de comunicación en tiempo real y enviarla al sistema de usuario final de destino para establecer la trayectoria de comunicación requerida. La figura siguiente representa el uso de un agente en señalización para establecer una conferencia multimedia, de esta forma el usuario B puede reservar los recursos de red necesarios y además puede cuidar de la configuración del sistema final para ambos usuarios.

Figura 22. Señalización futura basada en agentes móviles



Resumiendo, los agentes proporcionan nuevas oportunidades en el área de control de servicios, en particular la incorporación de agentes móviles dentro de IN para la obtención dinámica de servicios personalizados. Aquí los agentes móviles generados en el sistema de usuario final, serán enviados al SCP a fin de ejecutar manipulaciones específicas del perfil (por ejemplo la definición de una nueva tabla de enrutamiento). En el caso más extremo se puede utilizar esta aproximación para instalar nuevos servicios de usuario en el SCP, presumiendo que el SCP permitirá personalizar la lógica del servicio.

La integración de agentes móviles en propósitos de señalización no es sensible en IN, desde que la centralización del control del servicio representa la base de la arquitectura IN. Sin embargo las nuevas arquitecturas de Telecomunicaciones como TINA son los candidatos más apropiados para la incorporación de agentes móviles. Otras arquitecturas que promocionan los servicios basados en agentes móviles son AT&T y PersonaLInk.

5.3.3. Gestión basada en agentes

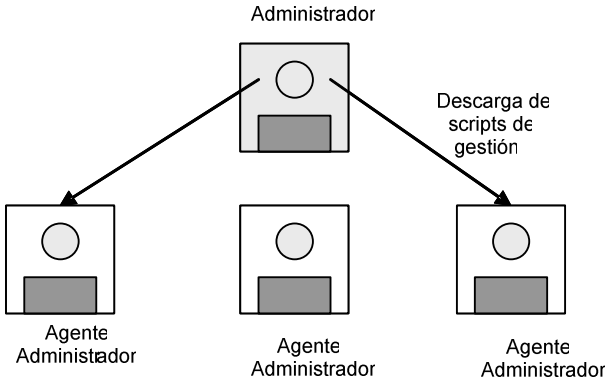
La distribución de tareas de gestión en redes ha sido investigada durante mucho tiempo, referido como Management by Delegation (MBD), adopta un paradigma de gestión descentralizado que aprovecha el incremento de la red computacional en los nodos de la red y la presión sobre los sistemas de red de gestión centralizados y el ancho de banda de la red. MBD permite la distribución temporal y espacial.

Los agentes de gestión ayudan a reducir la cantidad de comunicación entre el administrador y el agente de gestión soporta ejecución remota y manejo de scripts, que pueden ser activados mediante una base de tiempo, por acciones de gestión o por la ocurrencia de eventos específicos en el agente de gestión.

En este contexto el estándar de gestión OSI se define como una función de gestión de secuenciadores de comandos que permiten delegar actividadesde gestión desde el administrador hacia los agentes de gestión.

La figura siguiente representa esta aproximación en la que un agente de gestión baja el mismo script de gestión para múltiples agentes de gestión

Figura 23. Delegación de Inteligencia de gestión



Esto se puede realizar definiendo los scripts correspondientes de las operaciones de manejo que serán ejecutadas por el secuenciador de comandos localizado en el agente de gestión, permitiendo la pre-programación o retardo de ejecución de las operaciones del sistema de gestión.

En conclusión, los agentes móviles ejercen especial influencia de telecomunicaciones futuras y en las aplicaciones de gestión puesto que representan una alternativa para el modelo de interacción tradicional cliente/servidor que aporta mayor flexibilidad para la creación de servicios dentro de la próxima generación de entornos de Telecomunicaciones.

6. UN MODELO NEGOCIACIÓN ADAPTABLE PARA EL COMERCIO ELECTRÓNICO BASADO EN AGENTES

En un sistema del multiagente los agentes intentan adaptarse al ambiente aprendiendo o por un proceso evolutivo, haciendo así una anticipación de la interacción con los otros agentes. El papel presenta a un modelo de la negociación adaptable que usa la retroalimentación, como redes neuronales artificiales como la capacidad de aprendizaje para modelar la otra estrategia de negociación del agente.

La adaptabilidad y encarnación son dos problemas importantes que necesitan enfocarse al diseño flexible de sistemas multiagente. La adaptabilidad permite la generación de modelo del proceso de la selección dentro del sistema y así los resultados en representaciones interiores que pueden indicar las interacciones exitosas futuras. Para un agente de software su “cuerpo” refleja la historia de su adaptación en un ambiente virtual, incluso los constreñimientos, del ambiente y las interacciones y métodos de tratar con el ambiente. En el contexto de un multi -se relacionan sistema del agente, las dos propiedades, adaptabilidad y encarnación.

El principal propósito de adaptabilidad en el caso de un guión de la negociación es mejorar la competencia de la negociación del agente, basado en aprender de sus interacciones con otros agentes. En otros términos, para obtener un acuerdo y en el futuro, un bueno o incluso, el mejor trato. En este sentido es bien conocido que pueden verse los sistemas multiagente como un medio para alcanzar el equilibrio. En los recientes años se dieron varias soluciones a la adaptabilidad de agentes de software de aplicación.

Entre éstos, la mayoría usa el caso de una negociación Q-aprendiendo y el aprendizaje de Bayesian. Usaremos un agente vendedor/comprador para modelar la estrategia de negociación de agente.

6.1. Aprendizaje en la negociación basada en agentes

Como la negociación es un proceso general que involucra la comunicación entre dos o más agentes que intentan encontrar una manera de satisfacer los requisitos del agente, separados mutuamente emprendiendo las acciones apropiadas. Normalmente, es de involucrar la negociación en rondas múltiples para intercambiar las propuestas.

En este contexto es importante alcanzar el mejor acuerdo, la solución que es el uso de un modelo de negociación adaptable. Aprender en un ambiente multiagente es complicado por el hecho que, cuando otros agentes aprenden, el ambiente cambia rapidamente. Cuando los agentes interactúan y apriendien simultáneamente, sus decisiones se ven afectadas y limitadas.

6.1.1. La Estrategia de negociación

En un sistema multiagente los agentes deben coordinar sus actividades u objetivos y cuando los conflictos se detecten ellos deben poder negociar. Los conflictos pueden ser el resultado de la disputa del recurso limitada simple a los problemas más complejos donde los agentes discrepan debido a las diferencias entre su especialización del dominio. La negociación involucra el determinar un contrato bajo ciertos términos y condiciones.

De la perspectiva del CBB (Comportamiento del Comprador Consumidor), es la fase de la negociación donde el precio u otras condiciones de la transacción son determinados. Por ejemplo, la negociación usada en el comercio incluye las bolsas de valores, casas de subasta de arte finas, las subastas de flores, y las tiendas electrónicas basadas en comisión entre otras.

Como una coordinación de la técnica, la negociación permite evitar deadlocks y livelocks en los sistemas multiagente. La negociación se logra a través del intercambio de propuestas entre agentes.

La negociación puede hacerse sobre un juego de problemas y una propuesta consiste en valorar a cada uno de esos problemas y se genera autónomamente la estrategia del agente. Una manera de reforzar la autonomía de agente en un ambiente dinámico como un mercado electrónico es dotar su arquitectura de aprender de las capacidades.

Supongamos que tenemos dos agentes A y B que están negociando un conjunto de problemas que son representados por el vector X. $Xa_{\rightarrow}bt = \{Xa_{\rightarrow}b^1, Xa_{\rightarrow}b^2,\ldots, Xa_{\rightarrow}b^t\}$. $Xa_{\rightarrow}b^t [j]$ es la propuesta hecho por el agente A al agente B en el tiempo t para el problema j este bajo negociación. Cada agente tiene una función de utilidad U^i, $U^i:S^i \rightarrow R$. S^i es el estado del espacio pertinente al agente i y R es el conjunto de números reales. El orden de función de utilidad es el estado de preferencia. Cada agente tiene algún conocimiento privado y un poco compartido. Por ejemplo, cada agente sabe sólo de su propia función de utilidad y puede observar las acciones del otro agente, es decir las propuestas. También, cada agente tiene algunas creencias sobre el estado que sería el resultado de realizar sus acciones disponibles. Un agente puede aprender de la observación de las interacciones con el otro agente.

Las propuestas hechas por un agente son generadas por su estrategia de negociación que es específico a cada problema bajo la negociación. La estrategia de la negociación es privada a cada agente. Consideremos el caso de dos agentes, vendedor y un comprador de los que están negociando el precio un producto específico. Cada agente sabe su propio precio de reservación, RP, que es cuando el agente esta deseoso de pagar o recibir el trato. Un posible trato sólo puede hacerse si allí existe una zona de acuerdo entre la reservación del precio de los dos agentes. Los agentes ni siquiera saben si hay una zona de acuerdo. La regla ideal para encontrar si hay que una zona de acuerdo es la siguiente:

si RPvendedor <= RPcomprador entonces

* La zona de aceptación existe
* Posible trato

de lo contario

* La zona de aceptación no existe
* No hay trato

Donde el precio del comprador RP es el precio máximo que el comprador está a dispuesto a pagar por el producto, y RP del vendedor es el mínimo precio que el vendedor aceptará recibir por el producto. La manera en la que cada agente realiza una propuesta por el precio del producto es llamada estrategia de precios.

Normalmente, no puede haber ninguna previsión si un acuerdo no existe entre la negociación de dos agentes, así la capacidad de aprendizaje ayudaría en un gran porcentaje del trato que pueda hacerse en un mercado electrónico. Un agente de aprendizaje toma su decisión basada en su propio precio de reservación, en su propio dominio de conocimiento, y en la historia de propuestas de precio del pasado que se intercambiaron. A través del aprendizaje del agente se formará un modelo más exacto del otro agente y así podrá anidar modelos de agentes.

6.1.2. Toma de decisión

Tomando su decisión, un agente racional debe tener en cuenta las probables decisiones de otros cuyas decisiones son a su vez contingentes en su propia decisión. Esto lleva al bien conocido retroceda al problema de suposición, donde no existe ninguna predicción exacta o expectativa segura sobre las opciones individuales que pueden producirse.

Por consiguiente, todos los modelos de negociación intentan evitar este dilema involucrando una negociación estratégica. Una solución es enriquecer al modelo de negociación integrando planeación de razonamiento de inteligencia artificial, basado en casos y otras técnicas de la teoría de decisión.

Otra solución es incluir una capacidad de aprendizaje dentro del modelo de negociación que puede combinarse con técnicas de la teoría de decisión. Las soluciones principales para las que se adoptaron la inclusión de una capacidad de aprendizaje en un sistema del multiagente es el refuerzo de aprendizaje con un especial énfasis en Q-aprender, aprendizaje Bayesiano y el aprendizaje basado en modelos.

6.1.2.1. Aprendizaje de refuerzo

El aprendizaje de refuerzo es una técnica común usada por agentes adaptables en sistemas multiagente y su idea básica es revisar las creencias y estrategias basadas en el éxito o fracaso de la actuación observada.
Q-aprender es un algoritmo particular de aprendizaje de refuerzo (un refuerzo incremental de aprendizaje) que trabaja estimando los valores de todos los pares de estado-acción. Un agente que usa un algoritmo de Q-aprendizaje selecciona una acción basada en la función de acción-valor, llamado Q-función. $Qj(s, a) = Qj(s,a) + \lambda(rj - Qj(s,a))$, donde λ es una constante, el rj es el premio inmediato recibido por agente j después de realizar una acción en un estado s.

La Q-función define la suma esperada del premio descontado lograda la ejecutando de una acción *a* en un estado *s* y determinando las acciones subsecuentes por la política actual Π. La Q-función que usa pone al día la experiencia del agente. La técnica de aprendizaje por refuerzo tiene que tratar con el dilema exploración - explotación. Algunas comparaciones experimentales entre varias estrategias del exploración/explotación son presentados mostrando el riesgo de exploración en sistemas multiagente. Se demuestra que los algoritmos genéticos pueden usarse en sistemas basados en clasificadores eficazmente para lograr soluciones más cercanas a las óptimas que Q-aprendizaje.

6.1.2.2. Aprendizaje Bayesiano

Normalmente, el comportamiento Bayesiano es considerado como el único comportamiento racional del agente, es decir el comportamiento aumenta al máximo la utilidad. El aprendizaje Bayesiano se construye en base al razonamiento bayesiano qué proporciona una inferencia probabilística. El algoritmo de aprendizaje bayesiano manipula probabilidades junto con observación de datos. Se presenta una decisión secuencial que hace ejemplo de la negociación llamada Bazar en que el aprendizaje se planea como un proceso de actualización con creencia Bayesiano. Durante la negociación, los agentes usan el marco Bayesiano para poner al día el conocimiento y creencia que ellos tienen sobre los otros agentes y el ambiente.

Por ejemplo, un agente (comprador/vendedor) podría poner al día su creencia sobre el precio de la reservación del otro agente (vendedor/comprador) basado en sus interacciones con el vendedor/comprador y en su conocimiento del dominio. La creencia del agente se representa como un juego de hipótesis. Cada agente intenta modelar a los otros de una manera recursiva, durante el proceso de negociación, y cualquier cambio en el ambiente, si es pertinente y percibido por el agente, crea que, tenga un impacto en la fabricación de decisiones subsiguientes del agente. Los experimentos mostraron que la mayor zona de acuerdo, los mejores agentes de aprendizaje usan la oportunidad. El marco Bayesiano se usa como la creencia subyacente que pone al día el mecanismo. Además de proporcionar técnicas de actualización eficaces, la creencia Bayesiana conecta una red de computadoras, oferta un idioma de modelado expresivo y permite la codificación fácil y flexible de conocimiento dominio-específico.

6.1.2.3. Aprendizaje basado en modelos

El acercamiento intenta reducir el número de interacciones de ejemplos necesarios para la adaptación, invirtiendo más recursos computacionales para un análisis más profundo del pasado, dado la experiencia de la interacción. El proceso de aprendizaje tiene dos fases:

(1) el agente de aprendizaje infiere a un modelo del otro agente basado en interacciones pasadas y

(2) el agente de aprendizaje usa al modelo sabio por diseñar la estrategia de la interacción más eficaz para el futuro.

En un aprendizaje basado en modelos la interacción entre los agentes puede verse como un juego repetido de qué los agentes aprenden a modelos de los agentes rivales para la explotación en encuentros futuros. Las estrategias podrían ser representadas por reglas de la producción o por cualquier otro esquema de la representación.

6.1.3. Modelo de negociación adaptable

Extendiendo al modelo de la negociación orientado al servicio con un componente adaptable que se lleva a cabo por red neuronal. Este modelo permite la interacción entre dos agentes, uno comprador y varios vendedores. El cual busca el mejor precio y entabla una conversación para ponerse de acuerdo sobre el precio y llegar a un acuerdo final.

61.3.1. Protocolos de interacción

- Un protocolo de interacción es una conversación, una secuencia predeterminada de mensajes intercambiados entre agentes
- Con un PI, un agente sabe que mensajes puede enviar a otro agente y los mensajes que puede recibir.
- De esta forma un agente podrá dialogar con diferentes agentes, dentro de diferentes protocolos, sin aumentar excesivamente la complejidad de su implementación.
- En una conversación habrá un agente que la inicie (Initiator) y otro que responde (Responder o Participant).

Protocolos estándar definidos por FIPA[93]:

FIPA-Request

FIPA-Query

FIPA-Request-When

FIPA-ContractNet

FIPA-Interacted- ContractNet

FIPA-Auction-English

FIPA-Auction-Dutch

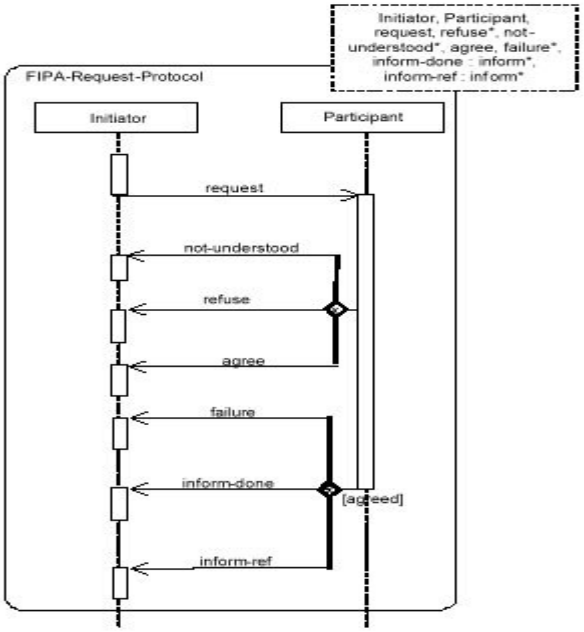
6.1.3.1.1. Protocolos de interacción - utilización

- Cuando un agente quiere utilizar un determinado protocolo en la conversacion con otro agente, debe rellenar el parametro: protocol del mensaje
- Un protocolo termina por dos razones:
    - Se llega al estado final de protocolo
    - Se elimina el nombre del protrocolo del parametro: protocol

93 http://www.fipa.org/

- Si un agente recibe un mensaje que intenta seguir un protocolo que no entiende, tendrá que devolver un mensaje de tipo refuse, explicando el motivo por el que se rechaza la comunicación.
- Si un agente que esta siguiendo un protocolo, recibe algun mensaje que no contempla ese protocolo, tiene que devolver un mensaje del tipo not-understood y para evitar caer en bucles infinitos, esta prohibido responder a un mensaje not-understood con otro mensaje not-understood

Figura 24. Protocolo de Interacción - Utilización



Permite que un agente le pida a otro que realice una operación (request). El participant puede aceptar (agree), rechazar (refuse) o no entender la petición (not-understood).

Si acepta la petición tiene que realizarla. Si no la da acabado informa de ello al Initiator con el mensaje failure. Si acaba tambien informa al initiator con un mensaje inform, que puede contener algún resultado, inform-ref.

Ejemplo del protocolo de Interacción – Utilización

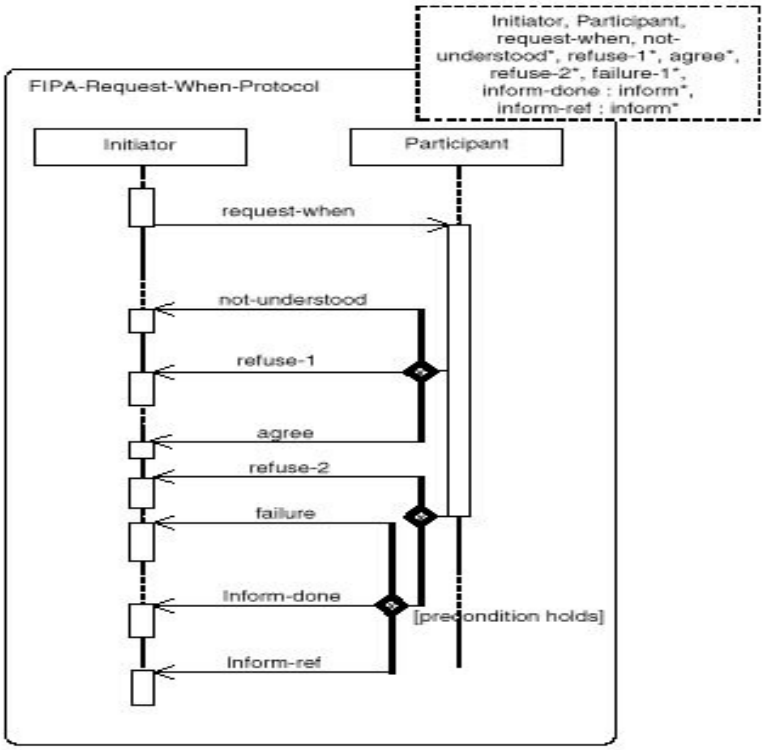
```
(request
  :sender Juan
  :receiver PEDRO
  :content   (action PEDRO(deliver regalo (location El Salvador)))
  :protocol FIPA-Request
  :reply-with order567)

(not-understood
  :sender PEDRO
  :receiver Juan
  :content  ((action PEDRO (rmequest  :sender Juan   :receiver PEDRO
              :content (action PEDRO(deliver regalo(location El Salvador)))
              :protocol FIPA-Request
              :reply-with order567))
              (unknown (word “regalo”)))
  :protocol FIPA-Request
  :in-reply-to order567)

(refuse
  :sender PEDRO
  :receiver Juan
  :content  ((action PEDRO(delimver regalo(location El Salvador)))(El Salvador is too far))
  :protocol FIPA-Request
  :imn-reply-to order567)

(agree
  :sender PEDRO
  :receiver Juan
  :content ((action PEDRO (deliver regalo(location El Salvador))) (priority order567 13:00PM))
  :protocol FIPA-Request
  :in-reply-to order567 )
(failure
  :sender revisor1
  :receiver Juan
```

:content ((action revisor1 ((arrive (plain)(at 8:00AM)))) (error “Acceso a Horarios”))
:protocol FIPA-Query
:in-reply-to r09)

(inform
:sender PEDRO
:receiver Juan
:content (deliver regalo in location El Salvador)))
:protocol FIPA-Request
:in-reply-to order567)

6.1.3.1.2 Protocolo de interacción solicita-cuando

Figura 25. Protocolo de Interacción Solicita-Cuando



Permite que un agente le pida a otro que realice una operación cuando se cumpla una precondición request-when. El participant puede aceptar (agree), rechazar (refuse) la petición o no entender el mensaje (not-understood). Si acepta la petición la realizará cuando la proposición se cumpla. Si al intentar hacerla falla, informa al Initiator con el mensaje failure. Si la acaba tambien informa al initiator con un mensaje inform, que puede contener algun resultado inform-ref. Si despues de acordar hacer la tarea, no es capaz de realizarla, envia un acto refuse.

```
(request-when
  :sender Juan
  :receiver megafonista1
  :content (action megafonista1 (call (Manolo)) (plane arrive at 8:00AM) )
  :protocol FIPA-Request-When
  :reply-with order567)

(not-understood
  :sender megafonista1
  :receiver Juan
  :content  ((action megafonista1 (request
                                    :sender Juan
                                    :receiver megafonista
                                    :content (action megafonista1 (call Manolo )
                                              (plane arrive at 8:00AM)))
        :protocol FIPA-Request-When
        :reply-with order567))
     (unknown (word “AM”)))
  :protocol FIPA-Request-When
  :in-reply-to order567 )

(refuse
  :sender megafonista1
  :receiver Juan
  :content  ((action megafonista1 (call(Manolo)) (plane arrive at 8:00))(not permited))
```

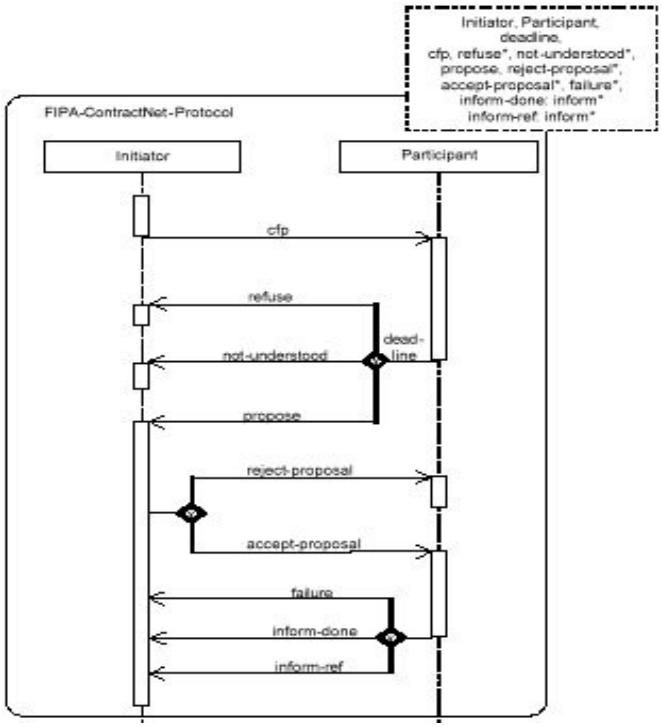
```
  :protocol FIPA-Request-When
  :in-reply-to order567 )

(agree
  :sender megafonista1
  :receiver Juan
  :content ((action megafonista1 (call(Manolo))(plane arrive at 8:00AM)) ())
  :protocol FIPA-Request-When
  :in-reply-to order567 )

(refuse (2)
  :sender megafonista1
  :receiver Juan
  :content
    ((action megafonista1 (call (Manolo)) (plane arrive at 8:00AM)))(change turn))
  :protocol FIPA-Request-When
  :in-reply-to order567 )

(failure
  :sender megafonista1
  :receiver Juan
  :content
    ((action megafonista1 (call(Manolo)) (plane arrive at 8:00AM))(megaphone damaged))
  :protocol FIPA-Request-When
  :in-reply-to order567 )
```

6.1.3.1.3. Protocolo de interacción contractNet

Este protocolo se usa cuando el agente Initiator quiere que uno o varios agentes realicen una tarea. Para ello pide propuestas para realizar la acción poniendo unas precondiciones en la ejecución de la tarea.

Los agentes receptores pueden rechazar la propuesta, no entenderla o devolverle a su vez otra propuesta al Initiator con sus precondiciones para realizar la tarea, como pueden ser: precio, instante de tiempo en el que la ejecutará, etc.

Figura 26. Protocolo de Interacción ContractNet



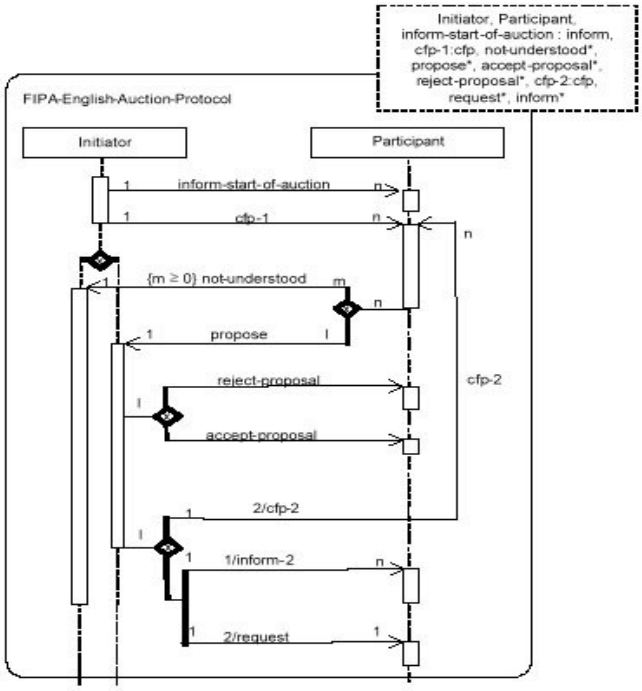
Cuando el Initiator recibe las propuestas, las evalua y escoge uno, varios o ningún agente para realizar la tarea. Envía la confirmación de la aceptación de las propuestas a los agentes elegidos y el rechazo de la propuesta al resto. Los agentes elegidos se comprometen a realizar la tarea, debiendo informar al Initiator al terminar la tarea del estado en que ha quedado está.

Como el Initiator tiene que esperar a que todos los agentes le respondan, no puede quedarse esperando por la respuesta de un agente que puede que este 'muerto', por esto se incluye un timeout (deadline), es decir, pasado un determinado tiempo desde que se lanzo la propuesta, el Initiator rechaza todas las que le llegan, informando al agente de que se rechazo porque se supero el timeout.

(cfp
  :sender Juan
  :receiver i, j, k, l
  :content
   ((action i, j, k, l(travel Ourense-París))(any ?x (and (= (price travel) ?x) (< ?x 90$))))
  :protocol FIPA-ContractNet
  :reply-whit ofer01
  :language FIPA-SL)

6.1.3.1.4. Protocolo de subasta-inglesa

Figura 27. Protocolo Subasta-Inglesa

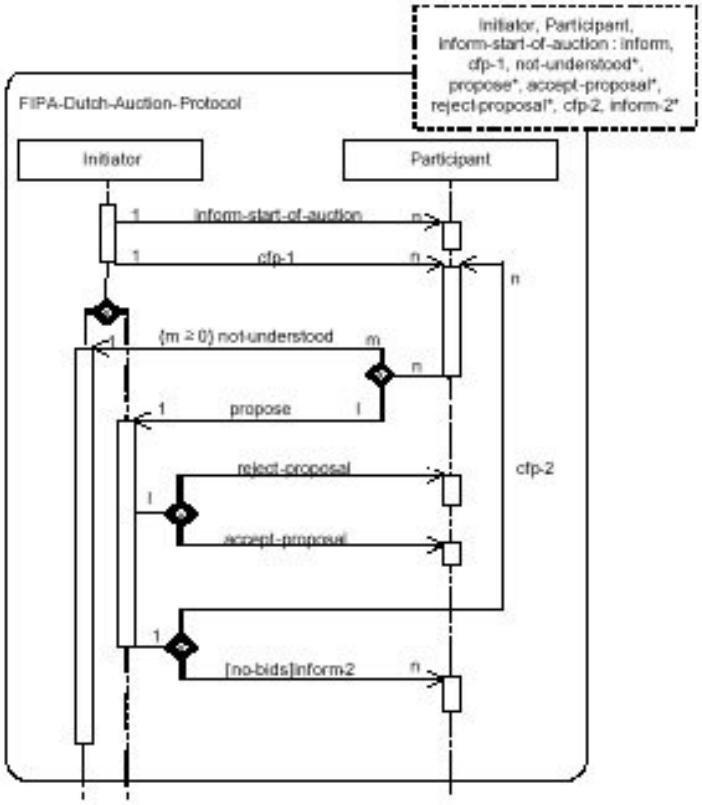


- Es un protocolo de “subasta”.
- En este protocolo un subastador intenta encontrar el precio de venta de un objeto, proponiendo inicialmente un precio por debajo del valor del mercado y despues ir gradualmente aumentando el precio.
- Al anunciar el precio, el subastador espera un tiempo para ver si algún comprador expresa su intención de pagar la cantidad.
- Si un agente acepta el precio, el subastador incrementa el precio y lo anuncia al resto esperando nuevas propuestas.
- La subasta termina cuando
    - No hay ningún comprador para el precio propuesto
    - Si un pujador supera el precio de reserva (privado) => se vende
    - Si la ultima oferta el precio es menor que el precio de reserva => no se vende

6.1.3.1.5. Protocolo de subasta-holandesa

- Es un protocolo de “subasta”.
- En este protocolo un subastador intentar encontrar el precio de venta de un objeto proponiendo inicialmente un precio por encima del valor del mercado y despues ir gradualmente reduciendo el precio.
- Al anunciar el precio, el subastador espera un tiempo para ver si algún comprador expresa su intención de pagar la cantidad.
- Si ningun agente acepta el precio, el subastador disminuye el precio y lo anuncia al resto esperando nuevas propuestas.

Figura 28. Protocolo Subasta-Holandesa



- La subasta termina cuando
  - Hay algún comprador para el precio propuesto
  - Se reduce el precio hasta un minimo => no se vende
  - Está inspirado en el funcionamiento de los mercados de flores holandeses.

6.2 Implementación de un agente en Java

Toda clase que implemente un agente móvil debe heredar de una clase genérica MobileAgent, que ofrece los siguientes métodos (ciclo de vida) que pueden redefinirse:

- inicializar(): se llama una sola vez, al crear el agente
- arrancar(): se llama cada vez que se arranca el agente (por ejemplo, al llegar a un lugar)
- parar(): se lama cada vez que se va a transferir, almacenar, o antes de eliminarlo
- concluir(): se llama sólo una vez, cuando el agente se destruye

Asimismo, un agente móvil tiene una identidad, que puede implementarse como una clase AgentIdentity y ofrece un objeto AgentInterface para comunicarse con otros.

6.2.1. Implementando ontologias con JADE

Si los agentes se comunican en cierto modo tiene que tener sentido para ellos, ellos deben compartir el mismo idioma, vocabulario y protocolos. Siguiendo las normas de FIPA, Jade apoya un cierto grado en común; Esto es evidente en el uso de actos comunicativos FIPA y las clases Coder/Decoder para idiomas de SLn que determinan la forma de los mensajes intercambiados entre agentes. Sin embargo, se necesitará definir su propio vocabulario y semántica para el contenido de mensajes intercambiados entre sus agentes.

Esto significa definir una ontología. De hecho, JADE proporciona tres maneras de implementar la comunicación entre agentes.

1. Primero y la manera más básica consiste en usar Strings para representar el contenido del mensaje. Esto es conveniente cuando el contenido de los mensajes es un dato atómico, pero no en el caso de conceptos abstractos, objetos o estructuras de datos. En tales casos, el String necesita ser parseado para acceder sus varias partes.
2. La segunda manera explota la tecnología de Java para transmitir objetos Java Serialized directamente como el contenido del mensaje. Éste es a menudo un método conveniente para una aplicación local donde todos los agentes se llevan a cabo en Java. Un inconveniente es que estos mensajes no son leíbles por humanos.
3. El tercer método involucra la definición de los objetos a ser transferidos como una extensión de una clase predefinida para que Jade pueda codificar y decodificar mensajes en un formato del estándar FIPA. Esto permite a agentes JADE la interoperabilidad con otros agentes del sistema.

Los tres tipos de contenidos de mensajes son reflejados por un Árbol de Dependencia en los tipos de contenido de los mensajes, los diferentes métodos que pueden ser usados para poner u obtener contenidos. La tabla de abajo da la correspondencia.

Tabla III. Correspondencia de métodos de obtención de contenidos

Content type	Getting content	Setting content
Strings	getContent()	SetContent()
Java Objects	getContentObject()	SetContentObject()
Ontology Objects	extractContent()	fillContent()

6.2.2. Interfaz gráfica de JADE

En el lenguaje de programación Java, un GUI corre en su propio hilo (el hilo evento-disparador) permitiéndole manejar y reaccionar rápidamente a eventos que se generan siempre que el usuario interactué con el GUI vía un componente como apretar un botón o redimensionar la ventana. Por otro lado, un programa de agente corre en su propio hilo de ejecución la cual le permite manejar su comportamiento. Porque no es eficaz permitir que un hilo llame directamente a los métodos del otro hilo, JADE ha proporcionado un mecanismo apropiado para manejar interacciones entre los dos hilos al integrar un GUI con un agente.

El mecanismo es simplemente basado en paso de eventos (event passing). Este mecanismo trabaja considerando las dos direcciones de interacción entre un GUI y un agente.

1. El agente interactúa con el GUI - Un GUI ya tiene un mecanismo built-in - de manejo de eventos que se implementa vía el método actionPerformed (...) de cada componente que es registrado con un objeto ActionListener.

   Para registrar un componente de su GUI con un objeto ActionListener, se hace su implementación GUI la interfase ActionListener y entonces registra todos los componentes interactivos de su GUI como lo son botones con este ActionListener vía el método addActionListener (...) o para cada uno de los componentes interactivos, se crea un objeto de ActionListener anónimamente y lo agrega al componente pasándolo en el argumento al mismo método addActionListener.

Siempre que una llamada al GUI se haga, un ActionEvent es generado por el componente de la fuente que invoca el método actionPerformed (). Y según el código que se proporciona dentro del método actionPerformed (), el GUI responde al proceso de eventos.

2. El GUI interactúa con el agente - JADE proporciona la clase abstracta GuiAgent que extiende la clase Agente. Esta clase tiene dos métodos específicos postGuiEvent () y onGuiEvent (). Éstos son los dos métodos que permiten manejar las interacciones entre un GUI y un programa de agente.

   Para poder usar estos métodos, el programa de agente debe extender la clase de GuiAgent. Entonces se debe proporcionar el código necesario dentro del método onGuiEvent () que el agente usará para recibir y procesar eventos que son anunciados por el GUI vía el método postGuiEvent. Se puede ver el método onGuiEvent () como el equivalente del método actionPerformed () en el GUI.

   Cuando un programa de agente que extiende la clase GuiAgent inicia, lanza un *comportamiento* específico - el GuiHandlerBehaviour - maneja eventos entrantes del GUI y los despacha a los manejadores apropiados y siguiendo exactamente el mismo mecanismo como en el GUI. Para anunciar un evento al agente, el GUI simplemente crea un objeto GuiEvent, agrega los parámetros requeridos y pasa en argumento al método postGuiEvent (). Puesto que este método pertenece a la clase GuiAgent, se necesita proporcionar al GUI una referencia a la clase del agente en la que el GUI puede invocar ese método.

6.2.3 Código de un modelo de negociación basado en agentes

En orden para ilustrar el uso de agentes, consideraremos una situación del comercio electrónico simplificada donde agentes compradores piden citas de agentes vendedores y entonces proceden a comprar al precio más barato - si está dentro del presupuesto; si no, ellos piden otra ronda de citas. Los Vendedores contestan demandas con un precio que es válido durante un tiempo limitado. Los Vendedores guardan etiquetas en qué precio ellos citaron a cada cliente y acepta una compra si el precio ofrecido es igual a la cantidad citada Y si la demanda se recibe a tiempo. Vendedores deben poder manejar varias solicitudes en paralelo. Planeamos que la transmisión tarda insertando retrasos entre la recepción de un mensaje y la contestación. Aquí esta un boceto de los intercambios de mensajes.

Figura 29. Intercambio de mensajes vendedor-comprador

En nuestro caso, hay varios problemas interesantes que tienen que ser resueltos.

1. Proporcionar interrupciones para limitar el tiempo gastado esperando por las respuestas.
2. cambiar comportamientos para lograr conversaciones coherentes

3. manejo de demandas paralelas
4. manejo de conversaciones paralelas
5. tratando con mensajes no deseados que se salen de la cola de entrada

6.2.3.1 Agente vendedor

Se muestra abajo el setup para agentes Vendedores donde se reciben mensajes de QUERY_REF y, para cada uno, nosotros creamos un objeto de la transacción que debería:

1. construir y planificar el comportamiento requerido para enviar una cita y esperar por una confirmación, y
2. atributos del comportamiento para compartir información. En particular, una parte importante de datos es el QUERY msg cuya conversationID servirán como un único identificador para la conversación.

Se crea una subclase SequentialBehaviour, para la transacción, no necesitamos muchas variables comunes: simplemente el mensaje original, msg, y el precio citado; los otros atributos, reply, convID y la plantilla no son estrictamente necesarios pero simplifican el código. La conversación consiste en 2 comportamientos:

- un DelayBehaviour para simular el proceso de tiempo [0-2 s] antes de enviar la cita
- un Receptor donde nosotros esperamos por un posible orden y estamos de acuerdo o se niega a dependiendo del precio

Éstos se crean en el método onStart () que es llamado una vez por Jade después de que el objeto es creado. Es el equivalente del setup() para Agentes o init () para Applets.

Codigo del agente vendedor

```
import jade.core.Agent;
import jade.core.AID;
import jade.core.behaviours.*;
import jade.lang.acl.*;
import java.util.*;

public class Seller extends Agent {
   private Float preciosa;
   int price;
   Random  rnd = newRandom();
   MessageTemplate query  = MessageTemplate.MatchPerformative
         ( ACLMessage.QUERY_REF );
   protected void setup() {
      // Obtiene los argumentas pasados durante la creacion de este agente
       int app=0;
       try{
       Object[] args = getArguments();
            if (args != null)
            {preciosa = Float.valueOf(args[0].toString());
            price = Integer.parseInt(args[0].toString());
            System.out.println("Este es el precio ofertado: "+ preciosa);
            app=1; }
       } catch (Exception e) { e.printStackTrace(); }
       if (app==0) price =50;
      addBehaviour( new CyclicBehaviour(this) {
         public void action() {
            ACLMessage msg = receive( query );
            if (msg!=null) addBehaviour( new Transaction(myAgent, msg,price) );
            block();
         }
      });
   }

   class Transaction extends SequentialBehaviour {
```

```
ACLMessage msg,reply ;
String     ConvID ;
int price;
public Transaction(Agent a, ACLMessage msg,int price) {
   super( a );
   this.msg = msg;
   this.price = price;
   ConvID = msg.getConversationId();
}
public void onStart() {
   int delay = delay = rnd.nextInt(9000 );   //una espera de 2 segundos
   System.out.println( " - " +myAgent.getLocalName() + " <- Recibio una consulta de " +
         msg.getSender().getLocalName() + ".  Responder Q. " + price + ".");
   addSubBehaviour( new DelayBehaviour( myAgent, delay) {
      public void handleElapsedTimeout() {
         reply = msg.createReply();
         reply.setPerformative( ACLMessage.INFORM );
         reply.setContent("" + price );
         send(reply);
      }
   });
   MessageTemplate template = MessageTemplate.and(
         MessageTemplate.MatchPerformative( ACLMessage.REQUEST ),
         MessageTemplate.MatchConversationId( ConvID ));
   addSubBehaviour( new myReceiver( myAgent, 2000, template) {
      public void handle( ACLMessage msg1) {
         if (msg1 != null ) {
            int offer = Integer.parseInt( msg1.getContent());
            System.out.println("Consiguiendo respuesa Q." + offer +
                  " de " + msg1.getSender().getLocalName() + " & mi precio es $" + price );
            reply = msg1.createReply();
            if ( offer >= price )
               reply.setPerformative( ACLMessage.AGREE );
            else
               reply.setPerformative( ACLMessage.REFUSE );
```

```
                    send(reply);
                    System.out.println("  == " +
                          ACLMessage.getPerformative(reply.getPerformative() ));
                } else {
                    System.out.println("Interrupcion!  cita  de  precio  Q." + price + "  de  " +
getLocalName() + " ya no es valida");
                }
              }
           });
       }
   }  // --- clase Answer ---
// ========== Metodos Utilzados ========================
//  --- generando IDs  de Conversaciones --------------------
   protected static int cidCnt = 0;
   String cidBase ;
   String genCID() {
      if (cidBase==null) {
         cidBase = getLocalName() + hashCode() +
               System.currentTimeMillis()%10000 + "_";
      }
      return  cidBase + (cidCnt++);
   } // Aqui se genera un ConversationID para las conversaciones

//  --- generando distintos generadores al Azar --------------------
Random newRandom() {return  new Random( hashCode() + System.currentTimeMillis()); }
// ----- limpiando agentes que tomaron mensajes viejos fuera de cola
   class GCAgent extends TickerBehaviour {
      Set seen = new HashSet(),
          old  = new HashSet();
      GCAgent( Agent a, long dt) { super(a,dt); }
      protected void onTick() {
         ACLMessage msg = myAgent.receive();
         while (msg != null) {
            if (! old.contains(msg))
               seen.add( msg);
```

```
            else {
               System.out.print("+++ Vaciando Mensajes: ");
               dumpMessage( msg );
            }
            msg = myAgent.receive();
         }
         for( Iterator it = seen.iterator(); it.hasNext(); )
            myAgent.putBack( (ACLMessage) it.next() );
         old.clear();
         Set tmp = old;
         old = seen;
         seen = tmp;
      }
   }

// ---------- Imprimiendo Mensajes ---------------------------------------
   static long t0 = System.currentTimeMillis();
   void dumpMessage( ACLMessage msg ) {
      System.out.print( "t=" + (System.currentTimeMillis()-t0)/1000F + " en "
            + getLocalName() + ": "
            + ACLMessage.getPerformative(msg.getPerformative() ));
      System.out.print( "  de : " +
            (msg.getSender()==null ? "null" : msg.getSender().getLocalName())
            +  " --> a: ");
      for ( Iterator it = msg.getAllReceiver(); it.hasNext();)
         System.out.print( ((AID) it.next()).getLocalName() + ", ");
      System.out.println( "  cid: " + msg.getConversationId());
      System.out.println( "  contenido : " +  msg.getContent());
   }    //imprimiendo mensajes
}  // ========= fin de la clase del vendedor ==========
```

6.2.3.2 Agente comprador

En el agente comprador, usamos al propio Agente como almacén del estado común y construimos la secuencia del comportamiento en el setup() donde también enviamos el mensaje que pone fuera del proceso entero: una transmisión QUERY_REF para el Vendedor que pide citas del precio. Los Comportamientos incluyen:

- receptores en Paralelo para tratar las respuestas de los Vendedores. Éstos tienen una interrupción de 1 sec.
- entonces un retraso al azar de (0-2s) antes de enviar una DEMANDA al postor más bajo
- el comporamiento final es un Receptor que espera por AGREE/REFUSE con la conversación común ID.

Usamos un número al azar para guiar nuestras decisiones y determinar retrasos. Hay una oportunidad buena que ningún Proveedor conveniente se encontrará como resultado, o que los mensajes llegan para ser aceptados demasiado tarde. En caso de fracaso, nuestro Comprador pone fuera de otra ronda de negociación llamando de nuevo a una simple setup.

```
import jade.core.Agent;
import jade.core.behaviours.*;
import jade.core.AID;
import jade.lang.acl.*;
import java.util.*;
import jade.domain.AMSService;
import jade.domain.FIPAAgentManagement.*;
import java.io.*;
import jade.gui.*;
```

```
public class Buyer extends Agent {
   Random     rnd  = newRandom();
   int        bestPrice = 9999;
   ACLMessage msg, bestOffer;
   Behaviour flusher;
   protected void setup() {
      if (flusher==null) {
        flusher = new GCAgent( this, 2000);
      }
      bestPrice = 9999;
      bestOffer = null;
      msg = newMsg( ACLMessage.QUERY_REF );
      MessageTemplate template = MessageTemplate.and(
            MessageTemplate.MatchPerformative( ACLMessage.INFORM ),
            MessageTemplate.MatchConversationId( msg.getConversationId() ));
      System.out.println("Agente Comprador : " + getLocalName() + " buscando ofertas.");
      SequentialBehaviour seq = new SequentialBehaviour();
      addBehaviour( seq );
      ParallelBehaviour par = new ParallelBehaviour( ParallelBehaviour.WHEN_ALL );
      seq.addSubBehaviour( par );
      //*****************************************
       //busca los agentes registrados en el sistema
       AMSAgentDescription [] agents = null;
      try {
         SearchConstraints c = new SearchConstraints();
         c.setMaxResults(new Long(-1));
         agents = AMSService.search( this, new AMSAgentDescription(), c );
      } catch (Exception e) {
         System.out.println( "Problemas buscando agentes en el AMS: " + e );
         e.printStackTrace();
      }
      //*****************************************
        for (int i = 0; i<agents.length; i++) {
         msg.addReceiver( new AID( "s" + i,  AID.ISLOCALNAME ));
         par.addSubBehaviour( new myReceiver( this, 1000, template) {
```

```
        public void handle( ACLMessage msg) {
            if (msg != null) {
                int offer = Integer.parseInt( msg.getContent());
                System.out.println("Solicitando cita para la oferta Q. " + offer +
                        " de " + msg.getSender().getLocalName());
                if (offer < bestPrice) {
                    bestPrice = offer;
                    bestOffer = msg;
                }
            }
        }
    });
}

seq.addSubBehaviour( new DelayBehaviour( this, rnd.nextInt( 2000 )) {
    public void handleElapsedTimeout() {
        if (bestOffer == null) {
            System.out.println("No se consiguio cita !");
        } else {
            System.out.println("\nMejor Precio Q." + bestPrice +
                    " de " + bestOffer.getSender().getLocalName());
            ACLMessage reply = bestOffer.createReply();
            if ( bestPrice <= 80 ) {
                reply.setPerformative( ACLMessage.REQUEST );
                reply.setContent( "" + rnd.nextInt( 80 ) );
                System.out.print(" Pedira a "+ reply.getContent());
                send( reply );
            }
        }
    }
});

seq.addSubBehaviour( new myReceiver( this, 1000,
        MessageTemplate.and(
        MessageTemplate.MatchConversationId( msg.getConversationId()) ,
```

```
            MessageTemplate.or(
            MessageTemplate.MatchPerformative( ACLMessage.AGREE ),
            MessageTemplate.MatchPerformative( ACLMessage.REFUSE ))) ) {
         public void handle( ACLMessage msg) {
            if (msg != null ) {
               System.out.println("Se Consiguio " +
                     ACLMessage.getPerformative(msg.getPerformative() ) +
                     " de " + msg.getSender().getLocalName());
               if( msg.getPerformative() == ACLMessage.AGREE)
                  System.out.println("  --------- Terminado ---------\n");
               else
                  setup();
            } else {
               System.out.println("==" + getLocalName()+" tiempo fuera ");
               setup();
            }
         }
      });
      send( msg );
   }
//  --- generando IDs de Conversaciones --------------------
   protected static int cidCnt = 0;
   String cidBase ;
   String genCID() {
      if (cidBase==null) {
         cidBase = getLocalName() + hashCode() +
               System.currentTimeMillis()%10000 + "_";
      }
      return  cidBase + (cidCnt++);
   }
//  --- generando distintos generadores al Azar --------------------
   Random newRandom() {
      return  new Random( hashCode() + System.currentTimeMillis()); }
//  --- Metodo para inicializar ACLMessages --------------------
   ACLMessage newMsg( int perf, String content, AID dest) {
```

```
    ACLMessage msg = newMsg(perf);
    if (dest != null) msg.addReceiver( dest );
    msg.setContent( content );
    return msg;
}
ACLMessage newMsg( int perf) {
    ACLMessage msg = new ACLMessage(perf);
    msg.setConversationId( genCID() );
    return msg;  }
// ----- limpiando agentes que toman mensajes viejos fuera de la cola
class GCAgent extends TickerBehaviour {
    Set seen = new HashSet(),
          old  = new HashSet();
    GCAgent( Agent a, long dt) { super(a,dt); }
    protected void onTick() {
       ACLMessage msg = myAgent.receive();
       while (msg != null) {
          if (! old.contains(msg))
             seen.add( msg);
          else {System.out.println("==== Vaciando Mensajes:");
             dumpMessage( msg ); }
          msg = myAgent.receive();
       }
       for( Iterator it = seen.iterator(); it.hasNext(); )
          myAgent.putBack( (ACLMessage) it.next() );
       old.clear();
       Set tmp = old;
       old = seen;
       seen = tmp;
    }}
```

CONCLUSIONES

1. La sociedad donde vivimos es una sociedad dominada por la información, que se ha convertido en la materia prima más valiosa. La aparición de agentes autónomos ha conllevado un avance significativo en el manejo de la información.

2. El desarrollo de agentes está ligado con lo que se conoce como web semántica, en la que todo lo que circule por la web será entendible no solo por el usuario sino también por la máquina. En este sentido, la clave para el éxito de un agente es el grado en el cual entiende la información que está manipulando.

3. El potencial de la tecnología de agentes es enorme y evidente, y su incorporación en los diferentes campos de aplicación supone una ganancia en eficiencia, ahorro de tiempo para el usuario al no tener que ocuparse de tareas rutinarias que esta tecnología permite automatizar.

4. Un agente puede ser usado de múltiples maneras en el medio empresarial actual, por ejemplo:
    - Procesamiento de tareas asíncrono y cooperativo.
    - Caracterización y configuración de servicios.
    - Uso de servicio instantáneo y negociación activa.
    - Descentralización de la gestión.
    - Comunicaciones inteligentes.
    - Almacenamiento de información y soporte para tipos de información dinámicos.

5. Los beneficios de utilizar una arquitectura informática basada en agentes son:
    - Desarrollo rápido.
    - Robustez y seguridad.
    - Rápida respuesta a los cambios de requisitos.
    - Mejora del desempeño mediante la ejecución de las tareas en diferentes ordenadores
    - Seguridad: El acceso de datos es controlado por los agentes de acuerdo a los privilegios de acceso, también la arquitectura de agentes es compatible con los estándares de seguridad de Internet.

6. El comercio electrónico es una realidad en Internet y cada vez es mayor el número de personas que realizan sus compras a través de la red. En este campo los agentes pueden prestar su servicio al comprador y al vendedor.
    - Al vendedor le ayudan a contactar con el cliente, a captar nuevos clientes y a crear sus perfiles para poder ofrecer un servicio más personalizado.
    - Al comprador le permite realizar búsquedas más personalizadas sobre el producto o servicio deseado.

RECOMENDACIONES

1. Los agentes deben diseñarse e implementarse sobre estándares que permita ínteroperar una gran variedad de aplicaciones, como FIPA (Fundación para agentes físicos inteligentes) que proporciona un estándar que constituye un marco de referencia para desarrollar sistemas basados en agentes.

2. Se debe utilizar ontologías para representar el conocimiento, ya que esto permite a los agentes manejar la misma interpretación o significado de un concepto, asegurándose que puedan comunicarse entre si, ya que el dominio del conocimiento debe de ser independiente de la implementación de agentes.

3. Se debe utilizar agentes sólo en aplicaciones donde se pueda obtener un beneficio notable sobre su uso. Una solución basada en agentes se hace más sensible cuando la aplicación puede explotar las capacidades de la arquitectura básica de agentes, es decir, deba operar autónomamente, deba comunicarse con otro agente, o deba supervisar el estado de su entorno y decidir acerca de sus propias actividades basándose en el estado de ese entorno.

4. En el diseño de una solución basada en agentes se debe seleccionar una arquitectura de agente tal que cada uno de los agentes ejecute tareas específicas. Estas comunidades o redes de agentes se utilizan para comunicarse unos con otros y cooperar en la resolución de problemas.

5. Se debe diseñar a los agentes con la capacidad de interactuar con las aplicaciones existentes, permitiendo utilizar sistemas de información diversa e incompatible a través de la red o del Internet, manipular la información y presentarla según las necesidades del usuario.

BIBLIOGRAFÍA

1. “Trends in distributed artificial intelligence. Artificial Intelligence Review”, Chaib-draa, B., Moulin, B., Mandiau, R., and Millot, P., http://www.neoyet.com/Agentes.htm, 1998

2. “Hierarchical Model Agent Architecture Multiagent Environment Communication Between Agents”, http://taylorandfrancis.metapress.com, 2006

3. “Knowledge Engineering Review” , Hyacinth S. Nwana, http://www.sce.carleton.ca, 2006

4. “Es un agente, o solo un programa?: Una taxonomia para agentes autonomos”, Franklin S., Graesser A., http://www.www.cs.umbc.edu/agents/web, 2001

5. “Una panoramica sobre diferentes tipos de agentes”, http://www.cs.umbc.edu/agents/introduction/ao/5.shtml, 2005

6. “Hacia una estandarización de los marcos de trabajo para Sistemas Multi-Agentes”, FLORES, Roberto, http://www.acm.org/crossroads/espanol/xrds5-4/multiagent.html, septiembre 2002.

7. “Agent Applications: Building Complex Information Systems “, http://www.agent.com/Documentation/WhitePaper/Applications.html, 2006

8. “Agentes Inteligentes”, http://www.red-Infotech.com, 2006

9. “Definition of Intelligent Software Agents “,
http://www.softagent.com/engl/ales.html, 2004

10. “UMBC Agent Web: A Panoramic Overview of Different Agents Types”,
http://www.cs.umbc.edu/agents/introduction/ao/5.shtml, 2005

11. “Intelligents Agents: An Emerging Technology for netx Generation
Telecomunication INFOCOM”,
http://www.research.att.com:80/~hgs/infocom96, 2003

12. “Applications of Intelligent Agents “,
http://www.cs.umbc.ed/agents/papers/papers.shtml, 2004

13. “Agent Applications: Building Complex Information Systems”,
http://www.agent.com/Documentation/WhitePaper/Applications.html,
2006

14. “Definition of Intelligent Software Agents”,
http://www.softagent.com/engl/ales.html, 2005

15. “Architecture for a Secure Agent-Based Website“,
http://www.artiscorp.com/Default.htm, 2003

16. “WBI (Web Browser Intelligent)”,
http://www.raleigh.ibm.com/pwasoft.htm, 2004

17. Iglesias Fernández, Carlos, “Fundamentos de agentes inteligentes”, http://www.gsi.dit.upm.es/~mast/ps/reportiad.ps.

18. “ARCHON (Architecture for Cooperative Heterogeneous ON-line systems)”, http://www.elec.qmw.ac.uk/dai/archon/test_1.html, 2006

19. “AgentWeb de UMBC”, http://www.cs.umbc.edu/agents, 2005

20. “Conector de MUD”, http://www.mudconnector.com, 2005

21. “Tennyson Maxwell Workshop”, http://www.tenmax.com/workshop.html, 2003

22. “Agents IBM”, http://www.raleigh.ibm.com/iag/iaghome.html, 2005

23. “Personal Agents”, http://www.yourcommand.com, 2005

24. “Especificación oficial XML v1.0 “, http://www.w3.org/TR/REC-xml, 2006

25. “The World Wide Web Consortium”, http://www.w3c.org, 2005

26. “Foundation for Intelligent Physical Agents”, http://www.fipa.org, 2006

27. “Estándar W3C DOM”, http://www.w3.org/DOM, 2006

28. “JADE programmer’s guide”, 2005

29. “Sistemas Multiagente, Universidad de Vigo”, http://trevinca.ei.uvigo.es/~pcuesta/sm/, 2005