INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

No. 77 **Serie:** ROJA **Fecha:** Febrero 2000

Un Modelo de Agentes en Espacios de Interacción y la Especificación de su Interprete

Jesús M. Olivares Ceja[1]
Ma. del Carmen Domínguez[1]
Araceli Demetrio Aguirre[1]
Adolfo Guzmán Arenas[1]

RESUMEN

En este trabajo se presenta un modelo orientado a generalizar las interacciones que se realizan entre agentes y en particular se muestran ejemplos tomados del comercio, para lo cual se desarrolla el lenguaje de interacción entre agentes (LIA) y finalmente se describe la arquitectura del interprete de LIA incluyendo los módulos orientados al manejo de eventos inesperados y la comunicación entre agentes que manejan ontologías diferentes.

Palabras Clave: Agentes, comparador de ontologías mixtas, espacio de interacción, inteligencia artificial, LIA, máquina de eventos inesperados, ontologías.

[1]Centro de Investigación en Computación del IPN, México

UN MODELO DE AGENTES EN ESPACIOS DE INTERACCIÓN Y LA ESPECIFICACIÓN DE SU INTERPRETE

Jesús M. Olivares Ceja, Ma. del Carmen Domínguez Ayala,
Araceli Demetrio Aguirre, Adolfo Guzmán Arenas.

Email: jesuso@acm.org, dminguez@hotmail.com, aracely74@popmail.com,
aguzman@pollux.cic.ipn.mx http://redicic.cic.ipn.mx/modelo/

INTRODUCCIÓN

Actualmente con el aumento en la complejidad de los sistemas de información se hace necesario contar con elementos de software para ayudar al usuario en el manejo de la información.

Las herramientas informáticas que brindan características de personalización al usuario, proactividad, autonomía y colaboración son los agentes de software, los cuales se caracterizan porque su comportamiento está guiado por propósitos que toman o les son indicados por el usuario. Existen diversos tipos de agentes, algunos son para interfaces de usuario, otros ayudan en el manejo de grandes volúmenes de información o en casos donde se realizan actividades repetitivas con algunos documentos, como ejemplo, borrar correos electrónicos pasados, o enviar un mismo mensaje a diferentes personas.

En algunos casos los agentes toman acciones en nombre de su usuario, por lo que debe tenerse cuidado al utilizarlos debido a que si los agentes cometen algún error o incurren en alguna falta, se adjudicarán a su propietario. Para protección de los agentes, de los usuarios y de las empresas existen diversas medidas de seguridad y privacidad.

Aunque existen muchas definiciones sobre el concepto de agente [Riecken 94c][Maes 96][Maes 97] [Riecken 95][Ye 97] aquí no se pretende llegar a un consenso, por lo que consideramos que un agente es un proceso asíncrono (o demonio), el cual puede ser parte de una colección de procesos similares, que tienen un fin común, y que están geográfica y temporalmente dispersos[Guzmán 98].

Las aplicaciones de los agentes son diversas, se encuentran en manejadores de correo electrónico[Maes 94], prototipos para el entretenimiento[Grant 97], diseño colaborativo[Edmonds 94], manufactura[Zita 97], enseñanza asistida por computadora[Selker 94] [Sanchez 97][Ayala 98], comercio electrónico[Chavez 97][Noriega 97], milicia[Amori 92], filtros de información[Baclace 92], extensiones de los sistemas basados en objetos a basados en agentes[Amandi 97][Conrad 97], programación de agentes sin lenguajes de programación[Canfield 94], aplicaciones en internet[Etzioni 94][Barret 97], ambientes virtuales[Tu 94][Maes 95], entre otras.

En algunos casos a los agentes de software se les intenta mostrar con características antropomórficas, como emociones[Bates 94], personalidad [Moon 96] [Lester 97] [King 96], y creatividad [Boden 94], con esto los investigadores pretenden acercar a los usuarios una forma

amigable y de ayuda en sus sistemas de información.

Los agentes interactúan entre ellos para alcanzar sus propósitos o para intercambiar conocimiento, información o recursos. Interactúan con sus usuarios para recibir sugerencias o instrucciones sobre lo que el usuario pretende que haga o para ver lo que hizo. Durante las interacciones intercambian conocimientos, recursos, creencias, acuerdos.

La comunicación entre agentes se ha estudiado siguiendo dos enfoques, primero, lenguajes imperativos, en los que el usuario le especifica al agente lo que debe hacer[Rus 97]; segunda lenguajes declarativos, que se utiliza para intercambio de información, en este sentido se ha desarrollado principalmente el lenguaje KQML[Finin 93][Finin 94].

En el área de agentes también se ha tratado el problema de comunicación entre agentes que manejan ontologías diferentes, en el proyecto Cyc[Guha 94]. En nuestro modelo se desarrolla el Comparador de Ontologías Mixtas (COM), para permitir la comunicación entre agentes que manejan diferentes ontologías haciendo uso de equivalencias entre árboles de conceptos.

En este documento se describe un modelo sobre interacción entre agentes que tienen propósitos y para lograrlos participan en espacios de interacción. Al tomar un papel un agente puede entrar a un proceso de planeación en donde en ocasiones requiere de llevar a cabo algunos otros papeles previos al que lo conducirá a lograr un propósito inicial. Durante la ejecución de los agentes pueden ocurrir eventos inesperados generados por la Maquina de Eventos Inesperados (MEI). Para describir a los agentes y sus interacciones se propuso un Lenguaje de Interacción entre Agentes (LIA) y su interprete multihilos.

En el capítulo I se explican las características del modelo de agentes y espacios de interacción y los casos de prueba utilizando ejemplos del comercio. En el capítulo II se especifica la sintaxis y semántica del lenguaje LIA y su interprete. En el capítulo III se desarrolla el Manejador de Eventos Inesperados, en el capítulo IV se explican los componentes del Comparador de Ontologías Mixtas.

1. MODELO DE INTERACCIÓN ENTRE AGENTES

El modelo que se propone permite estudiar las interacciones entre agentes considerando que cada uno tiene propósitos, recursos y características diferentes. Durante las interacciones los agentes intercambian recursos, información entre ellos para alcanzar sus propósitos. En un momento dado puede haber relaciones de colaboración en forma emergente entre los agentes.

Las características y recursos del ambiente en donde están los agentes y las interacciones se describen mediante variables globales y regionales. Las variables globales se pueden ver y accesar por cualquier agente. Las variables regionales son aquellas que pueden compartirse entre agentes que las declaran explícitamente. Algunos ejemplos de variables son:

- Reloj, (global) tiene el tiempo utilizado en la ejecución de las hebras de un modelo.
- AutosCiudadDF (regional), indica el número de automóviles que circulan en la ciudad D.F.
- AutosCiudadMty (regional), indica el número de autos que circulan en la ciudad de

Monterrey.

- MercadosDF (regional), lista de los nombres de los mercados que existen en la ciudad D.F.
- Articulo (regional), contiene un articulo que un agente trata de comprar y otro intenta vender.

Además de las variables, los componentes del modelo(figura 1.1) son agentes y espacios de interacción; los principales procesos son:

- Administrador, que se encarga del casamiento entre papeles, planificación y replanificación de papeles a tomar incluyendo la resolución de contradicciones,
- MEI, que es la Máquina de Eventos Inesperados,
- COM, el Comparador de Ontologias Mixtas.

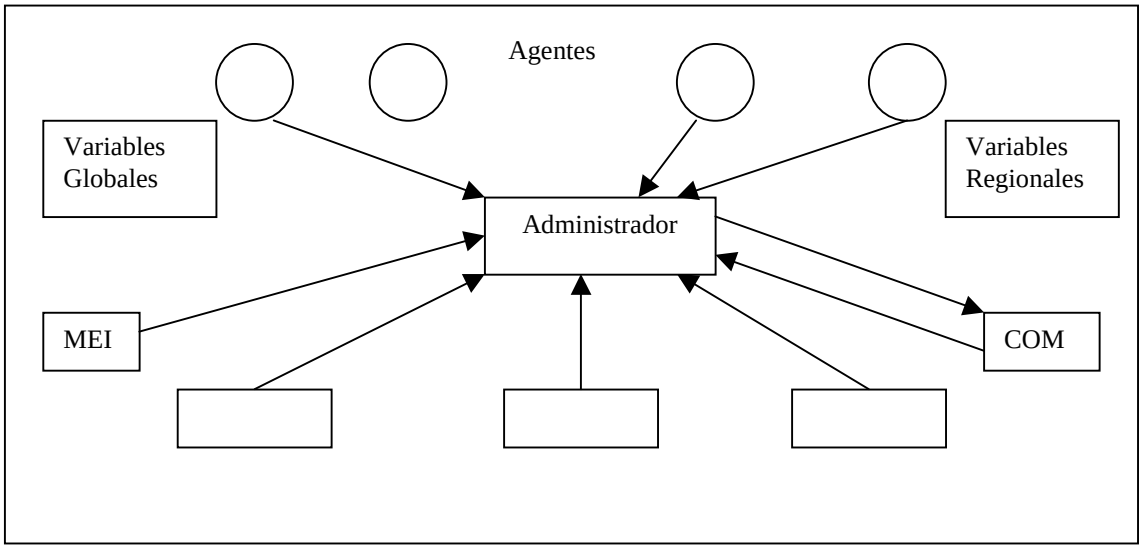

Figura 1.1 Componentes del modelo de interacción entre agentes con propósito

Los agentes son entidades que pueden llevar a cabo acciones de papeles con las que inician su ejecución en el modelo y las que adquieren de los papeles de los espacios de interacción.

Mediante la ejecución de los papeles el agente alcanza sus propósitos el agente puede ejecutar varios papeles a la vez, siempre que sean compatibles, por ejemplo, en el papel “comensal de un restaurante”, mientras ejecuta el papel de “vacacionista en CanCun”.

Por cada papel que adquiere el agente se deben cumplir los requisitos y restricciones que correspondan.

Cuando los papeles son contradictorios como en el caso de “viajero a París en avión”, que es contradictorio con “viajero a Los Ángeles en avión”, se replanifican los papeles en que participa el agente, con base en sus propósitos y sus prioridades.

Durante la ejecución de un modelo pueden ocurrir eventos inesperados que alteran la ejecución de los papeles y pueden cambiar el estado del modelo, por ejemplo cuando ocurre un terremoto.

El proceso de planificación, replanificación y atención de contradicciones toma en cuenta las prioridades de los propósitos de los agentes.

Cada vez que un agente toma un papel se genera un hilo de ejecución del mismo.

Los agentes se rigen por un mismo reloj global. El reloj avanza conforme se planifican instrucciones para que se ejecuten.

Los agentes perciben los eventos inesperados mediante una variable interna que indica los eventos que pueden percibir. Durante la ocurrencia de los eventos inesperados, algunos de ellos pueden provocar que el modulo Administrador se active para reasignar un plan de actividades a los agentes afectados; o sino hubo afectación, continua el curso normal de las hebras.

El usuario crea los ambientes de interacción indicando las propiedades tanto de las interacciones como de los agentes en un lenguaje llamado Lenguaje de Interacciones entre Agentes (LIA). Hay comandos de LIA para crear instancias de agentes y espacios de interacción.

1.1 AGENTES

Los agentes representan entidades físicas o lógicas del mundo real, por ejemplo: compradores, vendedores, licitadores, personas físicas y morales. Sus características, recursos y propiedades se especifican mediante variables internas que son propias de cada agente. Cuando se crea una instancia de un agente se heredan sus variables internas, sus características y sus propósitos a cada una de sus instancias.

Por ejemplo, el agente persona con propósitos comer y ser_rico; con variables internas, nacionalidad = “mexicana” y sexo = “masculino”, al crearse dos instancias del mismo, Juan y Pedro se obtiene:

- Juan (propósitos: comer, ser_rico) (variables internas: nacionalidad = “mexicana”, sexo = “masculino”)
- Pedro (propósitos: comer, ser_rico) (variables internas: nacionalidad = “mexicana”, sexo = “masculino”)

Cada propósito del agente tiene una prioridad que utiliza el modulo Administrador,

La información de las variables internas se comparte entre sus diferentes hilos. Por lo que si un hilo de un agente se entera que es viernes entonces para los demás hilos del agente esta información está disponible inmediatamente y no hay forma de que para otro hilo sea un día diferente, por ejemplo jueves.

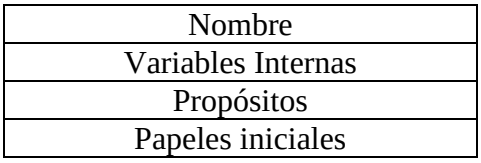

Figura 1.2. Componentes de un agente

Hay un papel inicial que permite asignar valores iniciales a sus variables internas.

1.2 ESPACIOS DE INTERACCIÓN

Los espacios de interacción (referidos aquí como interacciones) tienen papeles que pueden tomar los agentes para alcanzar sus propósitos. Los papeles tienen requisitos que los agentes deben cubrir para poder tomarlo. El atributo cupo del papel, indica el número de agentes que pueden tomar el mismo papel al mismo tiempo. Al tomar un papel, se inicia un hilo de ejecución que se asocia al agente que lo tomo. Un hilo tiene acceso a las variables globales, regionales, internas de su agente y a sus variables locales.

Papel 1 cupo	Papel 2 cupo	. . .	Papel N cupo
Nombre	Nombre	. . .	Nombre
Requisitos	Requisitos	. . .	Requisitos
Instrucciones	Instrucciones	. . .	Instrucciones
Variables Locales	Variables Locales	. . .	Variables Locales

Figura 1.3. Componentes de una interacción

1.3 CASAMIENTO ENTRE PROPÓSITOS Y PAPELES

Los agentes intentan alcanzar sus propósitos a través de llevar a cabo papeles que dan como resultado el logro de los mismos.

En algunas ocasiones al intentar alcanzar un propósito se debe elaborar un plan que incluye la ejecución de varios papeles que finalmente darán como resultado el que se alcance el propósito deseado. Este proceso puede generar varios caminos por lo que puede resultar un árbol AND-OR.

Antes de tomar algún papel, el agente debe de cubrir los requisitos que impone el papel en cuestión, para esto se utilizan sus variables internas, alguna globales y regionales como el tiempo, por ejemplo, para poder comprar el agente debe tener dinero. Además se deben de cubrir algunas restricciones que pueden ser de diferentes tipos:

- Físicas, tienen que ver con cuestiones de espacios y tiempo, por ejemplo si va a viajar a un lugar no debe estar viajando hacia otro en sentido contrario. En este caso el plan incluye que deberá elegir uno solo. Otro caso es cuando un agente quiere una pizza y tiene que volver a la oficina en 15 minutos y la pizza tardará 30 minutos, en este caso debe abandonar la pizza o faltar a la oficina.
- Personalidad, son aquellas que se relacionan con las actitudes del agente, por ejemplo, un agente tímido no podrá cruzar un río sobre una cuerda; otro ejemplo es que si es hombre no puede asistir a una escuela sólo para mujeres.

- Morales, son las que se relacionan con la calidad moral del agente, por ejemplo, una persona debe evitar dañar a otra.

Se tienen una estructura con los diferentes propósitos y los papeles que permiten alcanzarlos con el formato:

$$\text{Propósito} \rightarrow \text{Papeles}$$

por ejemplo:

- Necesita_Jarrito → comprador_subasta | comprador_mercado
- Comer → comensal_mercado | comensal_restaurante
- Ganar_Dinero → ingeniero | senador | arquitecto | abogado | vendedor_libros

1.4 Ejecución de Instrucciones

En circunstancias normales de ejecución de los papeles para alcanzar los propósitos de los agentes se toma la siguiente instrucción, cada instrucción se planifica insertandola en una cola de instrucciones que incluye la instrucción y el tiempo en que esta programada para ejecutarse.

Los eventos inesperados alteran el curso normal de ejecución de una hebra. Después de que se altera es posible que se dispare un proceso de replanificación. Los cambios que se producen como consecuencia de los eventos pueden ser:

- Positivos, por ejemplo cuando un agente encontró dinero y entonces ya puede ejecutar otras hebras.
- Negativos, por ejemplo, si un agente extravió dinero.

Después de que ocurre un evento inesperado se reanudan los hilos que no hayan cambiando.
Cuando se alcanza un propósito se termina los hilos que estén relacionados con el mismo.

Durante la ejecución, puede ocurrir que se generen excepciones que pueden originarse porque no se tiene acceso a algún recurso, se agotó algún recurso, o bien porque ha ocurrido un evento inesperado. Estas excepciones pueden dar origen a un proceso de replanificación. Esto también ocurre cuando el agente intenta alcanzar otro propósito.

1.5 Medición del Grado de Satisfacción de los Propósitos de un Agente

Como consecuencia de que en algunos casos los agentes logran alcanzar sus propósitos y en otras no, se propone una métrica para evaluar sus alcances. En una primera instancia se propone cuantificar el número de éxitos menos el número de fracasos en sus propósitos dividido entre el número total de propósitos que se buscaban alcanzar.

En una siguiente versión se propone asignar un peso a cada propósito de tal forma que ahora la ponderación es de acuerdo al peso relativo de los propósitos alcanzados.

Otra métrica propuesta es cuando un agente se compara con otros que son similares a él, es decir que sus variables internas son semejantes. Por ejemplo, un niño tiene 10 años y no va a la

primaria y se da cuenta que los demás si van, por lo tanto intentará ser como la mayoria. Otro ejemplo es el caso de un ingeniero que gana 5 mil pesos, cuando la mayoria gana 20 mil.

Una variante de la anterior es que el agente se compara con el promedio obtenido de varios agentes que son similares. Cuando un agente se detecta debajo del promedio se genera un estado de insatisfacción.

Otro caso que se considera es cuando un agente toma un papel adquiere ciertas expectativas de lo que va a realizar, por lo tanto despues de cierto tiempo se puede revisar si se alcanzaron las expectativas, por ejemplo, si una mujer toma el papel de secretaria, al final del dia se puede ver cuantas llamadas telefonicas realizó y cuantas cartas escribió, para autocalificarse respecto a pertinencia del papel que lleva a cabo.

Un aspecto a estudiar es la opinión de los demás respecto a un agente, si muchos agentes externos opinan que un agente es brillante, el agente receptor de las opiniones, puede comenzar a considerarse como brillante con base en la opinión publica. Las opiniones pueden ser positivas o negativas.

1.6 Casos de Prueba

El modelo de interacción entre agentes con propósito que presentamos en este documento, puede utilizarse en diferentes contextos, en esta sección se muestran casos tomados del comercio, el cual con el aumento de la población en el mundo conjuntamente con el proceso de globalización ha traído como consecuencia que se realice entre un número mayor de participantes.

Encontramos que es cada vez más cotidiano la celebración de acuerdos de libre comercio entre las distintas naciones del mundo. Por lo tanto somos participes de situaciones en donde las operaciones de compra-venta se realizan entre entidades de diferentes partes del mundo. En ocasiones las ventajas competitivas se logran al tener sucursales o representantes de la firma en el lugar de origen del demandante del bien o el servicio. de tal forma que la actividad tradicional de comercio ahora tiende a ser apoyada mediante el comercio electrónico.

El comercio electrónico, entendido como aquellas transacciones en las que no necesariamente se tiene una presencia física entre el ofertante y el demandante, ha tenido un gran auge desde sus inicios, a través de medios como el intercambio electrónico de documentos (Electronic Document Interchange, EDI) y la transferencia electrónica de fondos (Electronic Funds Transfer, EFT). Con la introducción de Internet se ha dado un mayor impulso al comercio electrónico, dado que ahora es posible publicitar un producto mediante presentaciones multimedia, aprovechando en parte las ventajas del protocolo HTML y el lenguaje Java. Se hablaba de que para finales de 1999, el mercado de Internet y subscriptores en línea se triplicaría de 6.8 millones en 1998 a más de 20 millones.

La población a la que puede accederse mediante el comercio electrónico basado en Internet es tan vasta que resulta humanamente muy difícil realizar el manejo de las operaciones manualmente, por lo anterior se hace necesario de la ayuda de herramientas informáticas que proporcionen confianza al proveedor pero al mismo tiempo que sean flexibles y atractivas para el consumidor, es por tanto deseable que sean personalizables al usuario debido a la gran cantidad de usuarios que se pueden encontrar en el mundo.

En este proyecto se propone modelar agentes que representen a compradores(demandantes), proveedores(ofertadores) para facilitarle al usuario un mudelo en donde puede estudiar las labores de comercio electrónico en Internet.

Para los casos que se proponen se han considerado los trabajos sobre agentes de comercio electrónico desarrollados en el MIT en el proyecto Kasbah[Chavez 1994], en donde un grupo de agentes que llevan a cabo transacciones de compra-venta, Los del proyecto FishMarket[Noriega 1998] el cual modela subastas de pescado, en donde los participantes (agente humano o de software) deben validar su información y su capacidad de pago antes de entrar a las subastas de pescado.

Los ejemplos que se desarrollan son acerca de compra, venta, subastas, ofertas y buscadores de ofertas.

1.6.1 CASO DE PRUEBA: SUBASTA

El lugar en que se hace la subasta los compradores pueden conocer los productos a subastar, o pueden pedir informes. El lugar es cerrado.

Los compradores se registran y demuestran liquidez como un deposito bancario o tarjeta de credito.

Cada comprador tiene una lista de los posibles productos a comprar ordenados por importancia. Cada uno tiene un precio mínimo, máximo y una tasa de incremento para llegar al maximo.

La subasta se inicia mediante un llamado que hace el subastador indicando el inicio de la misma.

Para cada producto subastable,
el subastador anuncia el producto a subastar indicando su precio inicial.

Cada comprador interesado por un producto emite su puja cuando su precio maximo es mayor que el precio de la ultima puja. Sus incrementos los hace en base a su tasa de incremento.

En cada puja, el subastador elige una puja que mejore la oferta antecedente, sino hay mejora, asigna el producto a la mejor postura hecha. Aunque puede suceder que nadie haya hecho alguna puja.

La descripción de la subasta en lenguaje LIA, entonces queda de la siguiente forma:

```
global
{
   int Reloj, tlimite ;
}

agent persona
  regional
  {
     char prod_subastandose[30] ;
     float ultima-puja:
  }
  intern
  {
     char nombre[30] ;
     double dinero ;
  }
  purpose
```

```
  {
     comer
     comprar_radio
   }

}
role subastador(instances 1)
  {
    requisite
    {
     buena_diccion
    }
    regional
    {
        int inicia_subasta ;
    }

    local
    {
     int i;
    }
     // INICIA LA SUBASTA
    out(inicia_subasta ,Reloj);
    iprod = 0;
    while( iprod < maxprod )
    {
        // ANUNCIA EL ARTICULO A SUBASTAR CON SU PRECIO INICIAL
        subastando=articulo[iprod];
        precio=precio[iprod];
        pujador=-1;
        out(subastando,Reloj);
        out(precio,Reloj);
        out(hagan_puja,Reloj+5seg);
        hay_pujas = true;
        while(hay_pujas)
        {
          out(revisa_pujas,Reloj+10seg);
          accept(revisa_pujas);
          ipujaador=1;
          haypujas=false; //suponemos que nadie mejora la puja
          while(ipujador <upujador)
          {
              if( oferta[ipujador] > precio)
              {
                precio=oferta[ipujador];
                pujador=ipujador;
                haypujas=true;
              }
                ipujador++;
          }
          if(haypujas)
            out(hagan_puja,Reloj+5seg);

        }
    if(ipujador > -1)
    {
     print("asignado a",ipujador);
     asignado=true;
     asignados[iprod] =true;
  }
```

```
}
 // DA POR TERMINADA LA SUBASTA
 out(fin_subasta,Reloj);
}
   papel comprador(cupo 10)
   {
    local
    {
     int i;
    }
    requisito
    {
        tener_liquidez;
    }
    accept(inicia_subasta );
    accept(subastando);
    accept(precio);
    accept(
    for( i = 0; i < max_compras; i++ )

      // si el producto subastado esta entre lo qquiere comprar
      //si precio < maximo
      //puja =  precio + incremento
      // EMITE PUJA
      // SE ENTERA DEL ESTADO DE SU PUJA
      // puja mientras no este asignado
      // cuando este asignado si es el mismo
      // paga y toma el articulo marcandolo como comprado
    }
   // itera mientras tenga algo porcomprar y este activa la subasta
}
```

1.6.2 Caso de Prueba: Compradores y Vendedores

El Comprador desea un producto que quiere comprar. El comprador debe tener dinero para tomar el papel. Busca lugares donde vendan el producto deseado, por ejemplo: zapatos en una zapatería, suéteres en una tienda de ropa.

Existen agentes vendedores que ofertan artículos que buscan los compradores, Esos agentes pueden buscar el articulo solicitado y ofrecerlo. El comprador puede esperar a que haya ofertas y entonces revisa el articulo, la calidad y el precio.

El comprador selecciona un articulo y emite la respuesta a quien le comprará el articulo (en la compra también se indico la forma en que el comprador puede pagar, efectivo, TC, deposito bancario, etc.) y el vendedor indico las formas en que le gustaría cobrar indicando sus preferencias de mayor a menor gusto. Entonces el comprador también indica la forma en que puede pagar.

Si se hace el pago, el vendedor recibe el pago y entrega el bien o le avisa al comprador que el bien ya va en camino en caso que lo haya colocado en algún servicio de mensajería.

Terminando el trato el agente comprador puede quedarse a revisar hasta que llegue el encargo o puede disparar otro agente en el que delega la autoridad para recibir el envío y el cual le avisara al agente comprador que se recibió su envío correctamente y que este bien.

Luego de que se recibe el bien, lo utiliza y si el bien NO cumple con las características puede haber una reclamación de la garantía con el vendedor quien debe atenderla y avisar si aplica o no la garantía.

1.6.3 Caso de Prueba: Ofertadores y Buscadores de Ofertas

Los ofertadores (pushers) son agentes se dedican a colocar ofertas a personas y agentes que son clientes potenciales de sus productos. Un ejemplo de estos son las personas que se dedican a repartir folletos a los transeuntenes de calles concurridas, otro ejemplo son aquellos que depositan en sus buzones de correo electrónico información de ofertas y promociones.

Los buscadores de precios y ofertas(pullers) son agentes que se dedican a solicitan precios, productos, condiciones de pago, garantías a diferentes proveedores y entonces las trasporta y proveen a aquellos agentes que desean adquirir un bien en las condiciones más propicias para el mismo.

2. Lenguaje de Interacción entre Agentes

El Lenguaje de Interacción entre Agentes con Propósito (LIA) se utiliza para definir los ambientes que se crean integrando agentes, espacios de interacciones y la descripción de las relaciones entre ellos.

El interprete de LIA consiste de tres etapas.

- La primera, se traduce un ambiente descrito en LIA hacia directivas que utiliza un constructor de ambientes.
- La segunda, es un modulo que construye las estructuras de datos y organiza los papeles para que los agentes los puedan referenciar.
- En la tercera, se realiza la ejecución del ambiente descrito inicialmente y participan el selector de papeles y manejo de contradicciones (Administrador), el Manejador de Eventos Inesperados (MEI) y el Comparador de Ontologias Mixtas (COM).

Durante la ejecución del modelo se crean instancias de los agentes e interacciones. Las instancias de agentes en ejecución, intentan tomar papeles de las interacciones que les permitan alcanzar el máximo de sus propósitos.

Cada vez que un agente toma un papel, se inicializa el contador de instrucciones en la hebra de ejecución. Conforme se ejecutan las instrucciones se avanza el contador de instrucciones de la hebra hasta terminar la ejecución del papel hasta que por medio de otro hilo se alcanzó el propósito que mantenía activo al papel en cuestión.

Durante la ejecución de las instrucciones de una hebra puede ocurrir que se deba replanificar la misma debido a algunas condiciones en el ambiente de simulación o del agente y algunas contradicciones que pueden ocurrir en sus recursos.

Debido a la ejecución simultanea de varios papeles de los distintos agentes, el simulador es multihilos, y el mismo cuenta con instrucciones para manejo de las mismas.

El modelo de interacción entre agentes se describe como sigue:

Modelo → Globales Agentes Interacciones Inicio

Donde Globales define las variables globales a utilizar, Agentes define a los agentes, Interacciones define las caracteristicas de las interacciones e Inicio contiene las instrucciones de inicialización de variables globales y regionales y la creación de las primeras instancias de los agentes.

2.1 VARIABLES

El estado del ambiente, recursos y sus características se define con los valores de sus variables regionales y globales, su sintaxis es como sigue:

Globales → global { ListaVarGlobal }

ListaVarGlobal → VarGlobal , ListaVarGlobal | VarGlobal

VarGlobal → Var

Var → int id ; | char id[numero] ; | float id ;

Regionales → regional { ListaVarRegional }

ListaVarRegional → VarGlobal , ListaVarRegional | VarRegional

VarRegional → Var

Las variables regionales y globales cuentan con un marcador de tiempo que indican el momento en que están disponibles.

2.2 AGENTES

La sintaxis para definir agentes en LIA es como sigue:

Agentes → Agente Agentes | Agente

Agente → agente id { VarInternas Propositos Papeles }

VarInternas → interna { LisVarInternas }

LisVarInternas → VarInterna , LisVarInternas | VarInterna

VarInterna → Var

Propositos → proposito { Proposito }

Proposito → id

Requisitos → requisito { ListaRequisito }

ListaRequisito → Requisito ListaRequisito | Requisito

Requisito → id

Papeles → Papel Papeles| Papel

Papel → papel id {VarLocales Requisitos Instrucciones }

Instrucciones → Instruccion Instrucciones | Instrucción

Instrucción → For | If | ExpresionAritmetica | InstruccionCompuesta

2.3 ESPACIOS DE INTERACCIÓN

La sintaxis para especificar espacios de interacción es como sigue:
Interaccion → interaccion id (cupo num) { Papeles }

VarLocales → VarLocal VarLocales | VarLocal

VarLocales → Var

2.4 INTERPRETE DE LIA

El interprete de LIA se utiliza para traducir a un formato ejecutable en computadora los ambientes descritos en LIA. Sus módulos principales son:

- El traductor, se encarga de tomar un archivo fuente con las descripciones del ambiente y produce un conjunto de directivas que sirven al constructor .
- El constructor, toma las directivas dadas por el traductor y establece las características de las estructuras de datos a utilizar durante la ejecución.
- El ejecutor, se encarga de coordinar el casamiento entre propósitos y papeles, la ejecución de las instrucciones de las hilos de los agentes y la coordinación de los eventos inesperados.

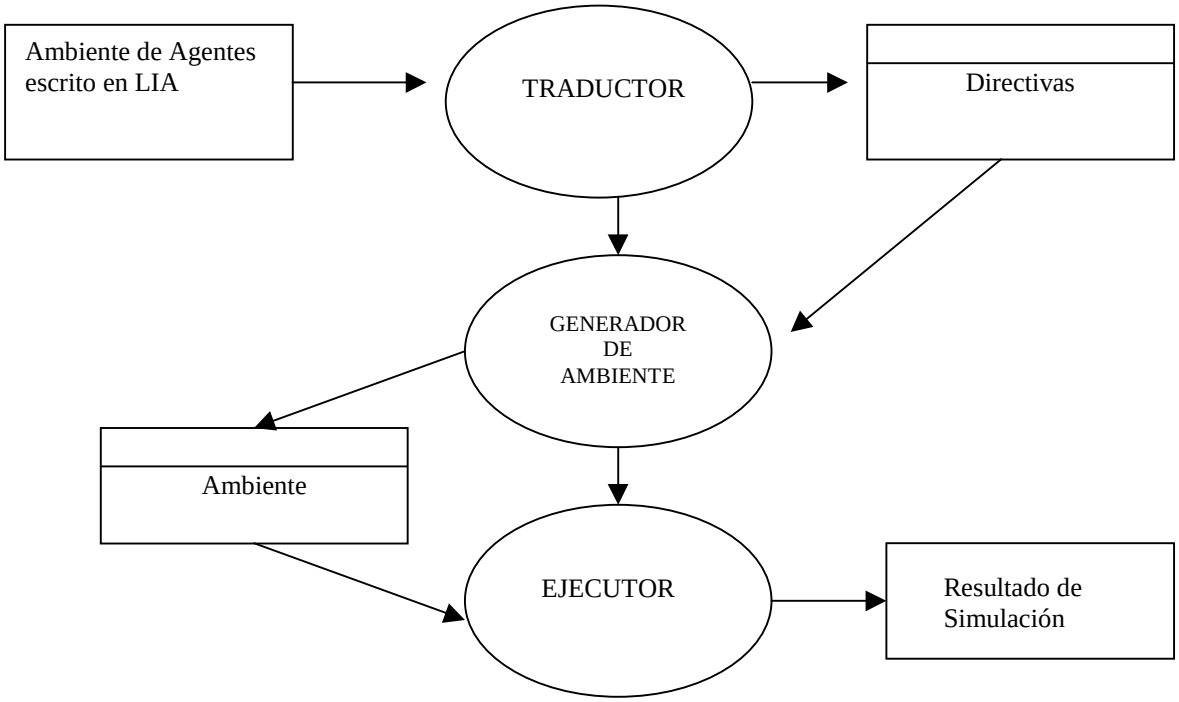

Figura 2.1 Diagrama de Flujo de Datos del Interprete de LIA

2.4.1 TRADUCTOR

El modulo traductor se encarga de tomar un archivo de texto escrito en LIA de acuerdo a las producciones gramaticales presentadas en 2.1, 2.2, 2.3 y 2.4. Ahí se definen las variables globales y regionales.

El traductor divide el archivo en unidades léxicas distinguiendo las palabras reservadas de LIA y los símbolos definidos por el usuario.

Utiliza la gramática de LIA con atributos semánticos para revisar sintácticamente y generar un conjunto de directivas que sirven para generar el ambiente.

Algunas directivas son:

- Variable Global idVarGlob
- Variable Regional idVarReg
- CreaAgente idAg
- Requisito idAg requisito
- Proposito idAg proposito
- Papel idAg
- Instrucción idAg idPapel instrucción

- Interaccion idInt
- Papel idInt
- Instrucción idInt idPapel instrucción

2.4.2 CONSTRUCTOR

El constructor toma las directivas del traductor y a partir de ellas crea las estructuras en memoria del interprete e inicializa la cola de instrucciones con los papeles iniciales de los agentes que las contienen.

El algoritmo de este proceso es el siguiente:

Mientras haya directivas

Lee directiva

Aplica directiva

2.4.3 EJECUTOR

El ejecutor inicia una vez que se encuentra cargado en memoria un ambiente que incluye variables globales, regionales, agentes, interacciones y se encuentran en la cola de instrucciones aquellas que corresponden con los papeles iniciales activos.

Su primera acción consiste en asignar los valores iniciales a las variables globales y regionales. Luego crea las instancias de los agentes e interacciones establecidas en el modulo de inicio del modelo.

Después de inicializar el ambiente, toma el reloj y comienza la ejecución de instrucciones contenidas en la cola. Cada instrucción después de efectuarse hace que se pase a la siguiente instrucción.

Cuando una instrucción ejecutándose corresponde a un evento inesperado, se suspenden las hebras, se da paso a la ejecución de los hilos correspondientes al evento inesperado, permitiéndole la modificación de las variables apropiadas. Al terminar el evento inesperado invoca al módulo Administrador para que evalúe los papeles en ejecución y detecte algunas variaciones que pueden ocasionar replanificación en algunos agentes y la terminación de algunos otros por satisfacción de sus propósitos o debido a que ya no puede continuar por falta de recursos o bien porque se considera que el agente ha “muerto”.

El ejecutor coordina los llamados al módulo COM cada vez que se requiera durante la ejecución del modelo cuando los agentes manejan ontologías mixtas.

Durante la ejecución se despliegan los eventos que van ocurriendo, por ejemplo, al crearse una instancia de algún agente o interacción, cuando se alcanza algún propósito, o cuando ocurre un evento inesperado.

El interprete cuenta con una línea de comandos donde el usuario puede suspender la ejecución del modelo y solicitar que se incluya la instancia de algún agente, interacción o bien, solicitar la modificación de variables globales y regionales.

Cuando durante la ejecución se han agotado las instrucciones de los papeles a ejecutarse, entonces se activa el módulo para medir el grado de satisfacción de los agentes participantes.

3. MÁQUINA DE EVENTOS INESPERADOS

Los eventos inesperados son acontecimientos no previstos. En nuestro modelo se maneja lo inesperado en una interacción entre agentes mediante una Máquina de Eventos Inesperados (MEI), cuyo propósito es administrar la forma en que se producen dichos eventos y como los diferentes agentes que lo perciben pueden reaccionar ante la presencia de una situación no prevista.

Los eventos inesperados están organizados en un árbol o taxonomía (figura 3.2). El árbol de eventos inesperados es infinito, sin embargo se ha considerado una taxonomía a manera de ejemplo, la cual es flexible. Por lo tanto hay un número infinito de eventos inesperados pero un número finito de reacciones que especifican que hacer cuando ocurre algún evento no previsto.

3.1 ELEMENTOS DE MEI

El sistema de Manejo de Eventos Inesperados consta de 3 elementos:

- Generador de Eventos inesperados (GEI): es un módulo que genera y envía los eventos inesperados al ambiente descrito en el lenguaje LIA, en el cual se encuentran las interacciones entre agentes. Se generarán eventos inesperados en dos formas: Aleatoria (el programa determina el evento, la fecha de ocurrencia y su duración) y Explícita (el usuario especifica cual evento, la fecha y duración).

  Los eventos inesperados que se hayan generado, son enviados al intérprete de LIA y a su vez, estos son clasificados en una taxonomía o árbol de eventos inesperados. (véase figura 3.1).

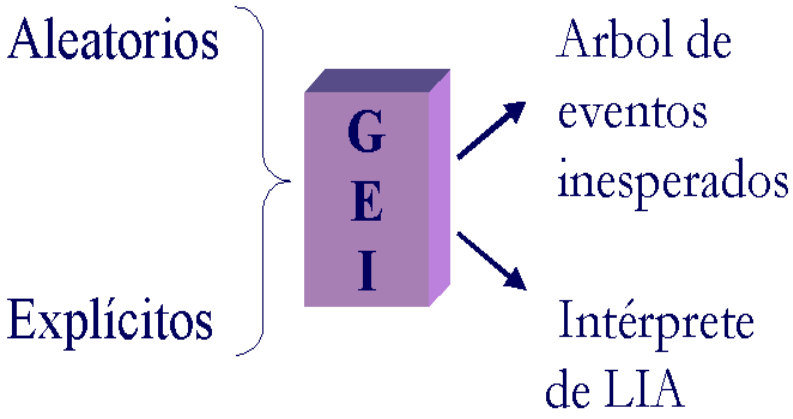

Figura 3.1. Generador de Eventos Inesperados GEI

Los elementos del evento son: nombre del evento, fecha de inicio y finalización del mismo[Paton 1999] y tiempo de duración. Los eventos inesperados se insertan en la cola de la vida especificada en LIA.

La cola de la vida, es una especie de contenedor de los eventos inesperados, así como la lista de las acciones (instrucciones en LIA) calendarizadas en el futuro. Después de que los eventos inesperados son generados, se insertan en dicha cola de la vida con un tiempo de ejecución asignado y cuando se cumple la fecha especificada se activan los eventos y se interrumpen las hebras de ejecución de aquellos agentes que perciben el evento inesperado.

♦ Un conjunto de nodos código, es un conjunto de instrucciones, las cuales indican como reaccionar cuando se presenta un evento inesperado. Los eventos inesperados están clasificados en un árbol o taxonomía. Cada nodo del árbol contiene un Nodo Código.

El árbol de eventos inesperados es infinito, sin embargo se ha considerado una taxonomía a manera de ejemplo, la cual es flexible. Por lo tanto hay un número infinito de eventos inesperados pero un número finito de reacciones que especifican que hacer cuando ocurre algún evento no previsto. (véase figura 3.2)

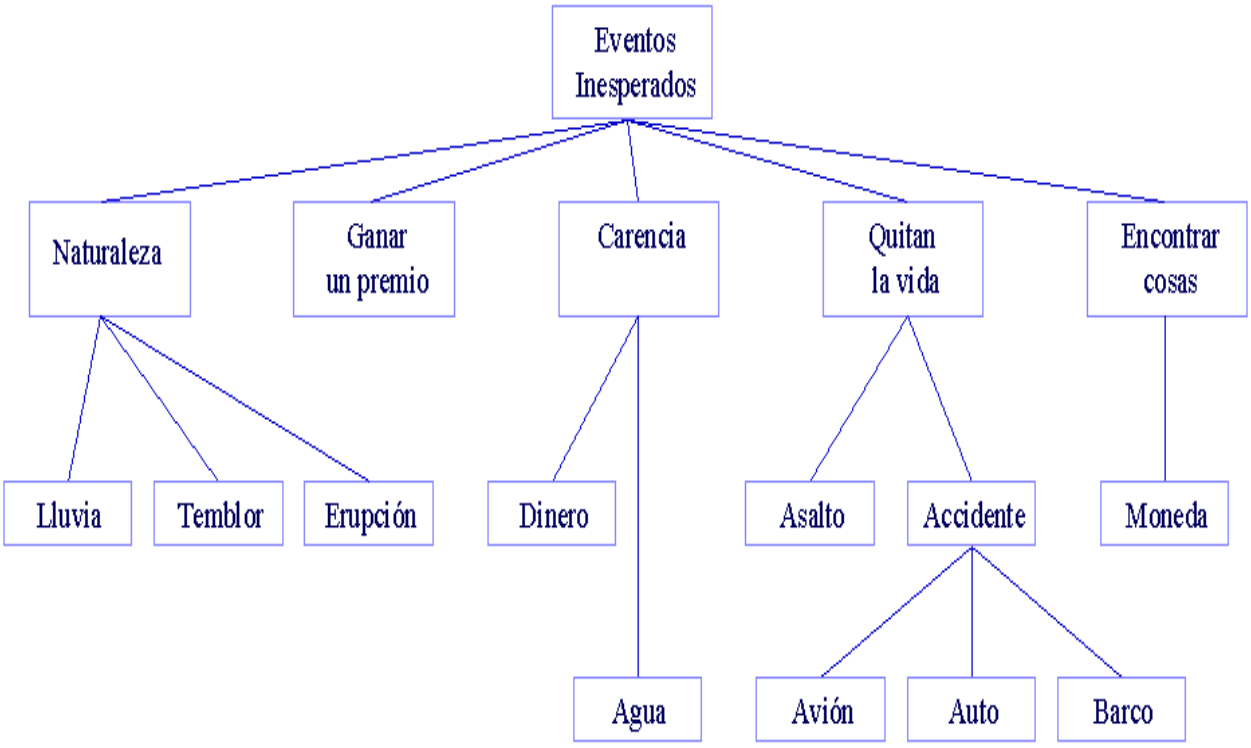

Figura 3.2. Árbol o Taxonomía de Eventos Inesperados

♦ Manejador de eventos inesperados (MAIN): Es el encargado de controlar y organizar todo el proceso que se genera cuando existe un evento inesperado en una interacción entre agentes.

Sus principales funciones son:

- Determinar cuales hebras de los diferentes agentes que se encuentran activos se suspenderán.
- Determinar cuales hebras se reanudarán al finalizar el evento inesperado.
- Realizar la búsqueda del evento inesperado activo en el árbol de eventos y determinar las hebras de reacción.
- Enviar las hebras de suspensión y de reacción al interprete de LIA.

3.2 FUNCIONAMIENTO

1.- GEI genera un conjunto de eventos inesperados, su calendarización y la información obtenida se almacena en GEI. Por lo tanto, GEI es el que tiene toda la información referente a los eventos que se van a enviar a LIA.

2.- Todos los eventos generados por GEI se colocan en la cola de la vida en forma de apuntadores, es decir, la cola de la vida contiene apuntadores a la próxima instrucción (de una hebra) a ejecutarse. Una variable global llamada EventoActivo es modificada en el ambiente de la interacción, para indicar que un evento inesperado está activo en tal momento. Para indicar la presencia de uno o más eventos inesperados, se le acumula 1 a la variable EventoActivo por cada evento que se encuentre activo y 0, cuando no exista alguno activo.

3.- Los eventos inesperados se clasifican en un árbol o taxonomía con sus respectivos nodos código, que indican la forma en como los agentes reaccionarán cuando un evento inesperado se active en una interacción entre agentes.

4.- MAIN es el encargado de la administración y control de los diferentes procesos que intervienen al presentarse un evento inesperado en LIA.

Tiene las siguientes funciones:

♦ Determinar cuales hebras de los diferentes agentes que se encuentran activos en una interacción serán suspendidas por la presencia de un evento inesperado. Para realizar tal actividad se tiene que considerar lo siguiente:

Cuando él interprete detecta un evento inesperado en la cola de la vida y si la fecha coincide con la que tiene él interprete, entonces éste debe activar el evento inesperado en LIA (los eventos generados están calendarizados con fecha y hora de ocurrencia).

Cuando LIA ejecuta un evento inesperado, el lenguaje llama a MEI con el argumento [evento]. Por ejemplo: MEI [rayo]. Y es hasta este momento cuando MEI entra en funcionamiento.

MEI se encarga de suspender las hebras que perciben al evento inesperado. Cada una de las hebras que tienen los agentes conocen qué eventos perciben, por lo tanto MEI solo suspenderá aquellas hebras que perciben el evento activo.

Además MEI tiene que saber qué agentes perciben qué evento, guardar el estado de las diferentes hebras para reanudarlas después de efectuarse el evento inesperado, interrumpir todas las hebras activas de los agentes que perciben el evento, determinar que hebras continúan al inicio del evento y determinar que hebras se reanudan al final del evento.

Por otro lado, MEI genera las siguientes estructuras de datos:

ArregloA: Esta estructura de datos contiene las hebras que se van a suspender, es decir son las hebras que perciben el evento y se encuentran activas en una interacción en LIA.

Aquí se considera que existen hebras que no necesitan suspenderse debido a que el evento inesperado no les afecta, por ejemplo: un agente x, está de vacaciones y una de sus hebras representa que esta comiendo en un restaurante y si en ese momento ocurre el evento lluvia, esta hebra no tiene porque suspenderse si la lluvia no impide a que siga comiendo, sin embargo si el evento se tratara de un terremoto, la hebra que indica que esta comiendo si se suspendería.(véase figura 3.3)

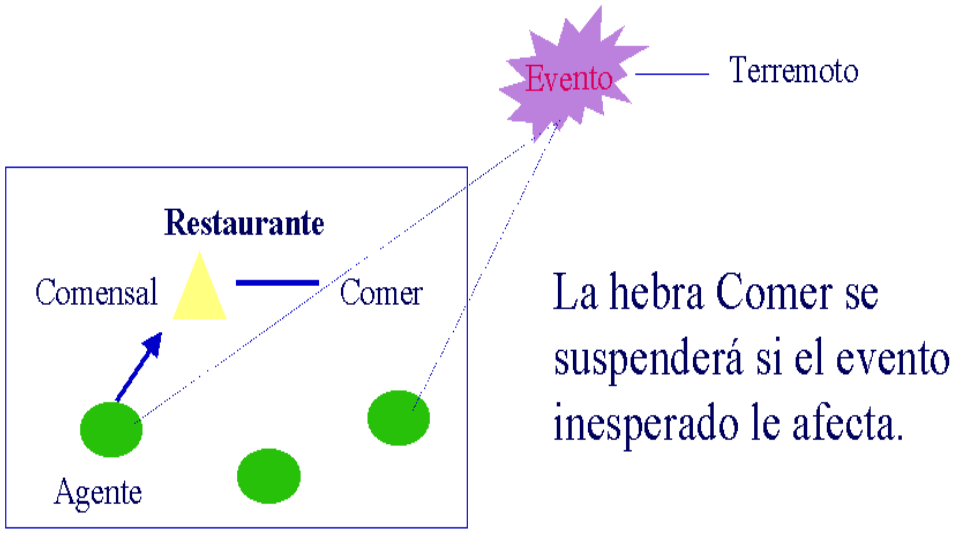

Figura 3.3. Evento terremoto suspende la hebra de ejecución comer.

ArregloB: Esta formado por las hebras de reacción, lo cual significa que contiene aquellas hebras que especifican lo que se debe hacer cuando ocurre un evento no previsto. Son las hebras adicionales a ejecutar en LIA.

Ambos arreglos se regresan a LIA, para que este realice la suspensión y ejecución de las hebras que se encuentren especificadas en dichos arreglos.

Cada reacción es un programa en LIA, se tiene que iniciar una hebra de emergencia para el evento que esta activo (véase ejemplo).

- Determinar cuales hebras se reanudarán al finalizar el evento inesperado. En este punto se tienen dos problemas específicos:

1.- Que no suspender

Cuando un agente participa en una interacción, adquiere un papel, que es el que desempeña, cada papel activo genera una hebra. Por lo tanto, se tienen varias hebras que representan diferentes actividades, en las cuales interviene tiempo, dinero o algún otro recurso.

Estos recursos se consumen cuando se participa en una interacción, por ejemplo se puede tener el papel de jardinero y un tiempo asignado a esa actividad, si un agente esta podando el pasto y ocurre un evento no previsto, por ejemplo cae granizo, se pueden generar diferentes comportamientos ante esta situación.

Uno de los comportamientos puede ser, caminar hacia un lugar techado y esperar a que deje de granizar. Después de que pasó el evento, se puede reanudar la actividad de podar el pasto, verificar que plantas se maltrataron con el granizo o tal vez hacer alguna otra actividad como ir a recoger a los niños porque ya es la hora de salida en la escuela.

La determinación de cual hebra suspender y cual no, depende de varios factores, en el ejemplo anterior se activará la hebra recoger a los niños, por la razón de que el factor tiempo indica que se lleve a cabo tal situación. (véase figura 3.4)

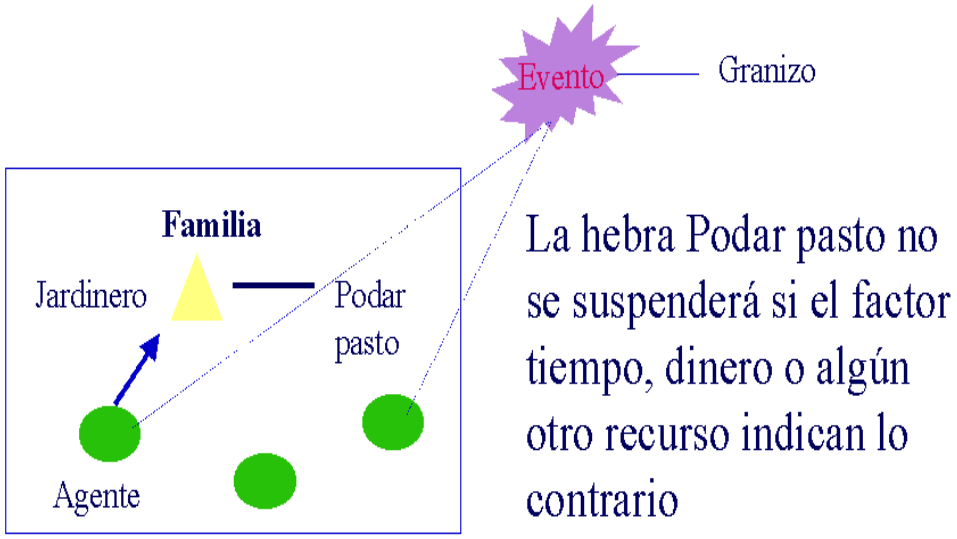

Figura 3.4. Evento granizo suspende la hebra de ejecución podar pasto.

2.- Que reacciones ignorar

Cuando un evento inesperado se encuentra activo, se generan hebras de reacción que especificaban como actuar ante el evento presentado. Sin embargo una vez terminado el evento no previsto, algunas hebras suspendidas se reanudan, otras hebras que estaban activas continúan su ejecución sin importar que el evento ya finalizó y otras que se generaron por la presencia del evento, se ignoran.

No todas las hebras de reacción se ignoran, esto depende de lo que esta haciendo el agente, por ejemplo, supongamos que un agente adquiere la hebra cocinar un postre y ocurre un incendio en su casa y sufre quemaduras de tercer grado, el agente toma la hebra estar en hospital, después de que el evento incendio ya finalizó, la hebra estar en hospital sigue activa a pesar de que el incendio ya haya terminado y la hebra cocinar un postre finaliza. (véase figura 3.5).

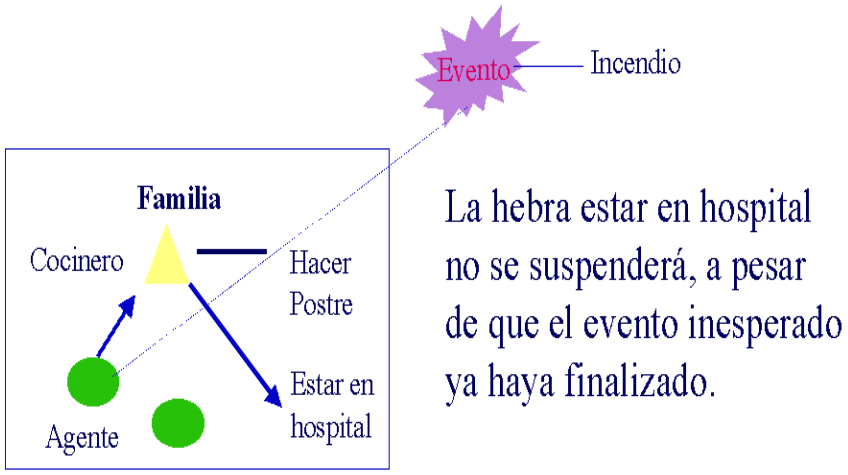

Figura 3.5. Evento incendio suspende la hebra de ejecución
hacer postre y permanece activa estar en hospital.

♦ Realizar la búsqueda del evento inesperado activo en el árbol de eventos(véase figura 3.2) y determinar las hebras de reacción.

Los eventos inesperados están clasificados en una taxonomía. El árbol de eventos inesperados es infinito, sin embargo se ha considerado una taxonomía a manera de ejemplo, la cual es flexible. Por lo tanto hay un número infinito de eventos inesperados pero un número finito de reacciones que especifican que hacer cuando ocurre algún evento no previsto.

Cada nodo del árbol contiene un conjunto de instrucciones que especifican que hacer cuando un evento inesperado este activo. A este conjunto de instrucciones le llamamos Nodos Código. Un Nodo Código es un programa que indica que acciones seguir cuando un evento no previsto este activo. Se tiene un Nodo Código por cada evento inesperado que se encuentre en la taxonomía. Hay un número infinito de eventos pero solo hay un número finito de programas.

Los Nodos Código están organizados en jerarquías (véase figura 3.2). Esto es con la finalidad de que no se tendrá un árbol global que abarque todos los eventos inesperados que pueden ocurrir, sino que tiene un cierto número de eventos que serán clasificados de una manera muy general, de tal manera que si un evento no esta especificado en el árbol o taxonomía, pero cumple ciertas características, el agente actuará como le indica el programa que se encuentra en el nodo de arriba del evento inesperado que se apegue más a las características de éste. (véase ejemplo).

En general, el árbol o taxonomía de los eventos inesperados contiene todas las reacciones (rutinas o programas) a todos los eventos inesperados de la gente. Todos los Nodos Código apuntan a la reacción más general, es decir al Nodo Código llamado EventosInesperados (figura 3.2), pero de acuerdo al evento inesperado que se encuentre activo, se ejecutará la reacción o rutina más específica a ese evento.

Esta rutina, es la que se regresa como hebra de reacción que es almacenada en el ArregloB del que se habló en párrafos anteriores.

La taxonomía de los eventos inesperados será asignada a nuestros agentes que se encuentran en LIA. Para ello, se han considerado tres posibles casos que están en proceso de prueba:

Caso1: Cada agente tiene todos los comportamientos

Caso2: Cada agente tiene alguno de los comportamientos asignados en forma arbitraria.

Caso3: Cada agente tiene algunos de los comportamientos asignados según sus características, por ejemplo si el agente es tímido, valiente, etc.

- Enviar las hebras de suspensión y de reacción al interprete de LIA.

Las hebras de suspensión y de reacción resultantes fueron asignadas en dos estructuras de datos, ArregloA y ArregloB respectivamente. MEI tiene la función en este punto de enviar dichas estructuras a al interprete de LIA, para que éste pueda realizar la suspensión de las hebras , así como activar las hebras de reacción (figura 3.6).

Figura 3.6. Hebras de reacción y suspensión se envían al intérprete de LIA

En la figura 3.7 se muestra de manera general el funcionamiento del Manejador de Eventos Inesperados (MAIN).

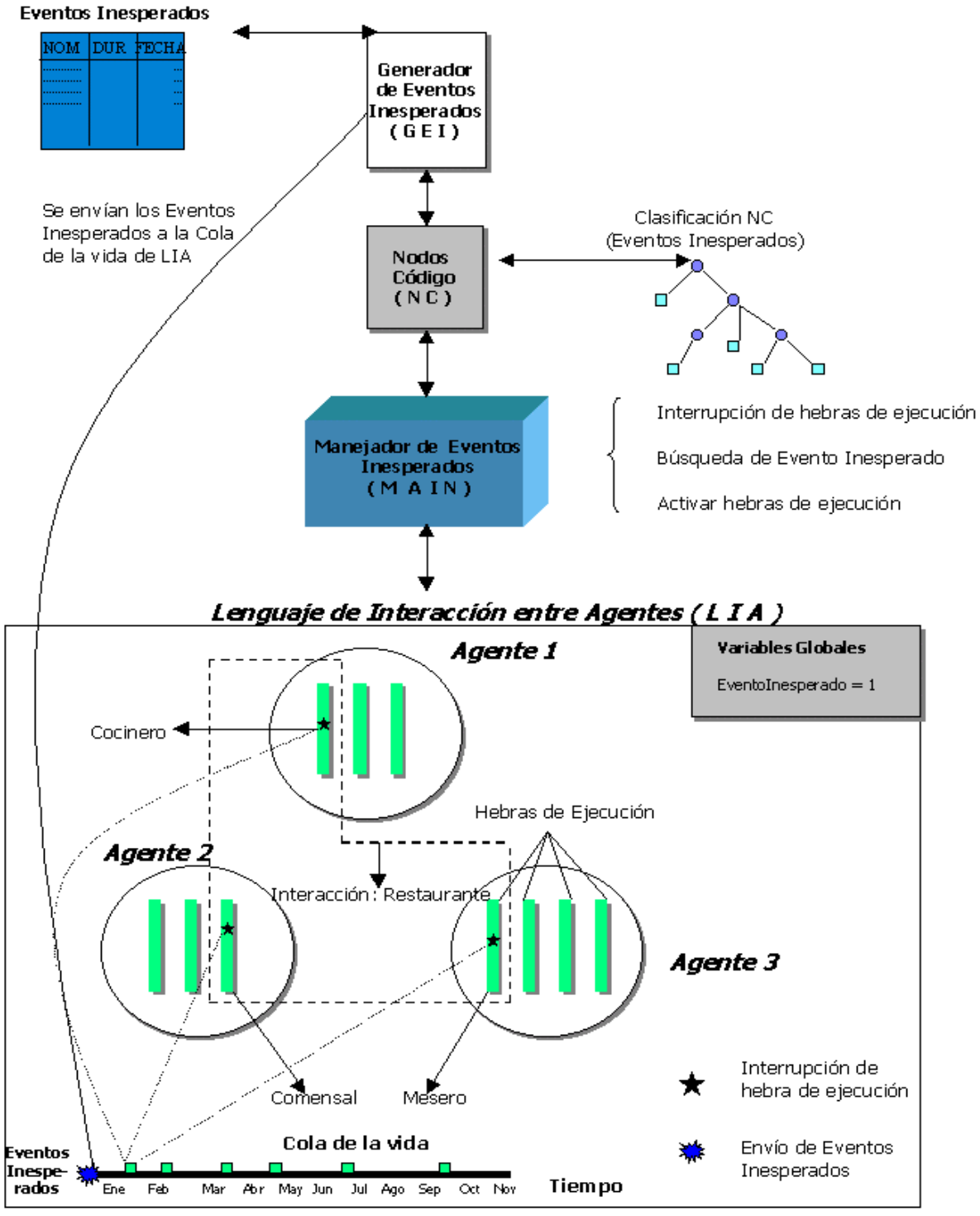

Figura 3.7. Manejo de Eventos Inesperados

El manejo de eventos inesperados, es uno de los aspectos que un agente debe saber para generar posibles cursos de acción y que de acuerdo a sus metas, sea capaz de tomar decisiones y ejecutar acciones acordes a sus metas.

3.3 EJEMPLO

Los agentes que participan en el modelo de interacción entre agentes con propósito son atómicos y se componen de varias hebras que representan los papeles que deben desempeñar, los cuales toman de las interacciones en las que participan. Una interacción contiene los papeles que desempeñan los participantes de la interacción, una vez que un agente toma un papel, adquiere una hebra de ejecución. Los agentes que participan en una interacción por ejemplo: Restaurante se encuentran activos en el Lenguaje de Interacción entre Agentes (LIA).

Por ejemplo, un agente X "comensal" puede interaccionar con un agente Y "mesero" en el script o interacción Restaurante(véase figura. 3.8).

Figura 3.8 Un agente tiene un propósito y toma un papel
en una interacción entre agentes

Los papeles o hebras de ejecución de los agentes(véase fig. 3.9) que participan en la interacción Restaurante son:

Agente 1: Mesero

Maestro de Ingles

Portero de un equipo de fútbol

Agente 2: Comensal

Hijo de familia

Arquitecto

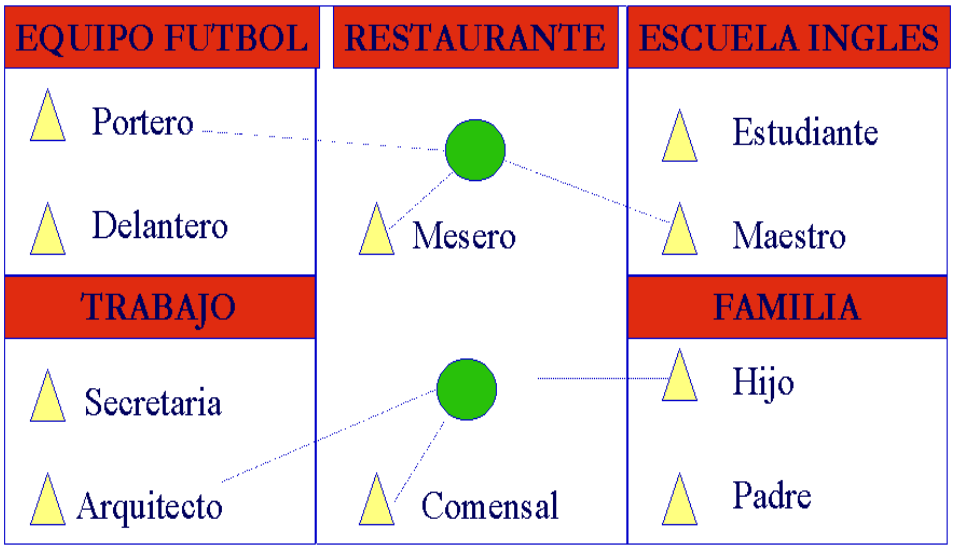

Figura 3.9. Hebras de ejecución de los agentes que participan en la interacción Restaurante

Al cabo de cierto tiempo, el comensal pide la cuenta, antes de que el mesero le entregue la cuenta, ocurre un evento inesperado: "terremoto" (véase figura.3.10), todos los agentes que intervienen en dicho script no saben como actuar ante esta situación, en ese momento el Manejador de Eventos Inesperados se activa e interrumpe todas las hebras de ejecución de la interacción(es decir, las hebras de ejecución de los diferentes agentes: comensal, mesero, cajero, cocinero y otros agentes que estén activos en la interacción Restaurante).

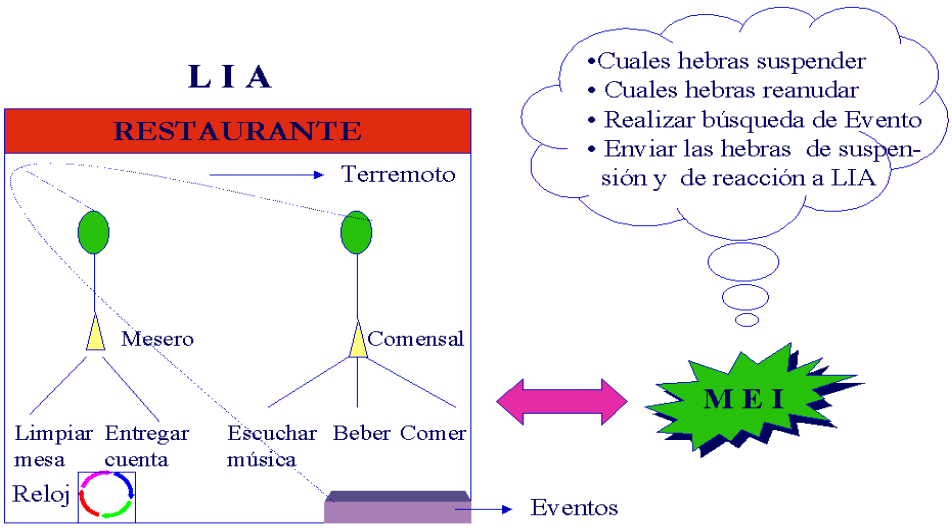

Figura 3.10. Presencia de un evento inesperado "terremoto" en LIA.

Para efectuar la interrupción de las hebras de ejecución de la interacción que esté activa, MEI tiene que considerar lo siguiente:

- Saber qué agentes perciben qué evento.
- Guardar el estado de las diferentes hebras para reanudarlas después de efectuarse el evento inesperado.
- Interrumpir todas las hebras que estén activas.
- Determinar que hebras continúan al inicio del evento.
- Determinar que hebras se reanudan al final del evento.

Una vez que las hebras fueron suspendidas, el evento inesperado activo se busca en un árbol de conceptos (que se encuentra clasificado de acuerdo a las características del evento, véase figura 3.2.), para saber cual va a ser el comportamiento del agente cuando el evento inesperado se presente (figura 3.11).

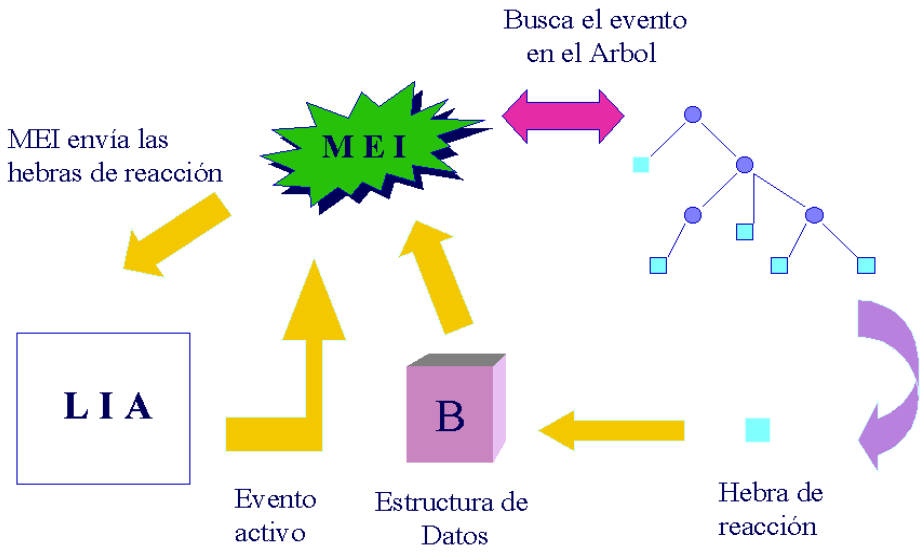

Figura 3.11. Búsqueda del evento inesperado cuando se encuentra activo en LIA

Los nodos código son instrucciones que van a indicar cual va a ser la reacción del agente dependiendo de varios factores, por ejemplo si se tratara del evento "asalto", se tiene que considerar si trae dinero, si se comporta agresivo con el asaltante, etc. Algunos ejemplos de las instrucciones de los nodos código expresados en una forma natural se muestran a continuación.

Asalto

Si dinero > 0

dinero =0

si dinero = 0

agente = sufre alguna lesión

si agente = ataca ladron

agente = muerto

Terremoto

Si ubicación = lugar techado

Agente = salir del lugar

En caso contrario

Agente = buscar proteccion

Incendio

Si agente = sufre alguna lesion

agente = pedir ayuda

En caso contrario

agente = permanece donde está

Finalmente la reacción que determine el nodo código, MEI la retorna a LIA para que éste active nuevas hebras de reacción. Al finalizar el evento inesperado, MEI determina cuales hebras se reanudan, cuales se ignoran o continúan activas a pesar de que el evento inesperado esté terminado y nuevamente se regresa el control a LIA en el cual se encuentran los agentes que participan en la interacción restaurante.

4. COMPARADOR DE ONTOLOGÍAS MIXTAS

4.1 DEFINICIÓN DE ONTOLOGÍA

Una ontología es una especificación explícita de una conceptualización. Lo que incluye es un vocabulario de términos y especificación de su sentido. El grado de formalidad de esta especificación va desde: muy informal, semi-informal, rigurosamente formal.

El término es tomando de filosofía, donde Ontología se refiere a existencia. Para I.A., lo que "existe" es aquello que puede ser representado. Un agente se va a asociar a una ontología.

El término ontología, en el ámbito del intercambio de conocimiento se usa para significar una especificación de una conceptualización. Esto es, una ontología es la descripción de los conceptos

y relaciones que pueden existir para un agente o para una comunidad de agentes. Un compromiso ontológico es un acuerdo para usar un vocabulario, que es consistente, aunque no completo, respecto a la teoría especificada en la ontología. Se trata entonces de definir un vocabulario común en donde se representa el conocimiento compartido.

Podemos construir agentes comprometidos con ontologías, y diseñar ontologías con las que los agentes pueden compartir conocimiento. Si algún agente entiende la ontología puede usar la información.

Las ontologías en sí mismas también pueden ser reutilizadas. La reutilización de algún pedazo de conocimiento requiere de una descripción explícita de los puntos de vista que están inherentemente presentes en el conocimiento. De otra forma, no hay forma de saber si el pedazo de conocimiento es aplicable a una nueva aplicación y porqué.

Una ontología especifica una conceptualización, una forma de ver al mundo. Por lo que cada ontología incorpora un punto de vista. Una ontología contiene definiciones que nos proveen del vocabulario para referirse a un dominio. Las definiciones dependen del lenguaje que usamos para describirlas.

4.2 CARACTERISTICAS TIPICAS DE LAS ONTOLOGÍAS

- Pueden existir ontologías múltiples:

El propósito de una ontología es hacer explícito algún punto de vista. A veces necesitamos combinar dos o más ontologías. Cada ontología introduce conceptualizaciones específicas.

- Podemos identificar niveles de abstracción de las ontologías.

Estos niveles de generalización nos dan una topología de ontologías. La idea es caracterizar una red de ontologías usando multiplicidad y abstracción. Como no podemos aspirar a tener una descripción completa del mundo, podemos pensar en una estrategia de construcción gradual de abajo hacia arriba.

- Multiplicidad de la representación.

Un concepto puede ser representado de muchas formas, por lo que pueden coexistir múltiples representaciones de un mismo concepto.

- Mapeo de ontologías.

Establecer relaciones entre los elementos de una o más ontologías, para establecer conexiones, especializaciones, generalizaciones, etc.

4.3 CONSTRUCCIÓN DE LA ONTOLOGÍA

- Captura:

(i) identificación de los conceptos y relaciones claves en el dominio de interés,

(ii) producción de definiciones no ambiguas de conceptos y de sus relaciones,

(iii) identificación de términos para referirnos a esos conceptos y relaciones

- Codificación: representación explícita de la conceptualización en un lenguaje formal:

(i) comprometerse a términos básicos de especificación (a veces se llama meta-ontología),

(ii) escoger el lenguaje de representación adecuado

(iii) codificarlo

- Integración de ontologías existentes: cómo, cuáles y si vamos a usar alguna ontología existente

4.4 LENGUAJES DE DEFINICION DE ONTOLOGÍAS

En principio podemos usar cualquier lenguaje de programación, pero a veces carecen de expresividad para escribir lo que queremos decir.

Algunos lenguajes utilizados para definir ontologías son: EXPRESS (STEP), CML (CommonKADS), Ontolingua. Estos incluyen primitivas como: constructores para agregados, múltiples jerarquías clase-subclase, reglas y axiomas; varias formas de modularización, para poder escribir diferentes ontologías y sus inter-relaciones; la posibilidad de tomar una visión a un meta-nivel. Lo que se busca en el lenguaje es expresividad y uso.

Aún no se tiene un estándar general, algunos propuestos son:

- Workflow management coalition (WfMC),
- STEP y su lenguaje de especificación EXPRESS Standard for the Exchange of Product model data: es una inter-lingua para definir y especificar productos, asociado a su ciclo de vida: diseño, manufactura, uso, mantenimiento y desecho. El objetivo es dar un mecanismo capaz de describir datos del producto durante su ciclo de vida.
- CORBA

The Common Object Request Broker Arquitecture está surgiendo como estándar para recuperar objetos y para invocar operaciones en objetos a través de la red.

Provee un mecanismo en donde los objetos pueden hacer peticiones y recibir respuestas de forma transparente.

El lenguaje (IDL - Interface Definition Languaje) especifica los objetos y las operaciones para aplicaciones remotas/distribuidas.

Incorpora nociones informales de ontologías.

- KIF y gráfos conceptuales

Knowledge Interchange Format y los grafos conceptuales son lenguajes para representar ontologías basados en lógica de primer orden.

KIF pretende ser un lenguaje capaz de representar la mayoría de los conceptos y distinciones actuales de los lenguajes más recientes de representación de conocimiento.

KIF está basado en lógica de predicados con extensiones pare definir términos, meta-conocimiento, conjuntos, razonamiento no-monotónico, etc.

4.5 INTEGRACIÓN DE ONTOLOGÍAS

Para la integración de ontologías se han desarrollado proyectos como CYC, el cual es un proyecto de MCC (Microelectronics and Computer technology Corporation) que da los fundamentos para razonamiento de sentido común mediante el desarrollo de ontologías para una gran variedad de aplicaciones específicas del dominio, todo el conocimiento esta representado declarativamente en una variante de lógica de primer orden en un lenguaje llamado: CYCL, tiene mecanismo de inferencia y de control, las ontologías están organizadas en conjuntos modulares llamados microteorías, c1ada microteoría captura el conocimiento y razonamiento requerido para un dominio específico, tales como espacio, tiempo, causalidad o agentes; pueden existir múltiples microteorías para un solo dominio reflejando diferentes perspectivas, se puede ver la ontología de CYC, más como una ontología monolítica, como una red de microteorías.

TOVE TOronto Virtual Enterprise: desarrollar una ontología para empresas, usa definiciones basadas en lógica de primer orden y permite deducir respuestas a preguntas de sentido común (usando Prolog).

Enterprise es un proyecto parecido a TOVE pero Inglés, el énfasis es sobretodo en proveer un ambiente de integración de herramientas y métodos usados en los negocios.

KACTUS es un proyecto ESPRIT para el desarrollo de una metodología de reutilización de conocimiento técnico, usa CML (Conceptual Modelling Language) desarrollado como parte de KADS dentro del proyecto de CommonKADS, hace distinciones explícitas entre conocimiento del dominio, de inferencia, de tareas y de resolución de problemas, una parte central es la biblioteca de ontologías organizadas por los niveles de abstracción.

Plinius, es un proyecto que tiene como objetivo, la extracción semiautomática de conocimiento a partir de textos en lenguaje natural enfocado a materiales cerámicos, usa un lexicón para mapear tokens de lenguaje natural a expresiones formales en el lenguaje de representación de conocimiento, la ontología define el lenguaje en que la parte semántica esté expresada.

4.6 COMPARADOR DE ONTOLOGÍAS

En nuestro modelo, el intercambio de información entre los agentes se puede realizar utilizando una ontología común, es decir, entienden los mismos conceptos; sin embargo, en la comunicación entre agentes muchas veces se utilizan palabras o expresiones que alguno de ellos no entiende, ya sea por que usan un vocabulario distinto, utilizan palabras que significan lo mismo, o utilizan las mismas palabras que tienen significados diferentes, por lo tanto se incluyo dentro de nuestro

modelo un programa que compare y maneje Ontologías Mixtas, lo que llamamos Comparador de Ontologías Mixtas.

El comparador de ontologías parte de que cada agente, en el sistema utiliza una ontología en la que se especifica su conocimiento. El problema se presenta cuando dos o más agentes requieren intercambiar información pero cada uno maneja una ontología diferente, por lo tanto se recurre al Comparador de Ontologías (COM), al cual se le especifica el concepto a traducir y la ontología en la que se encuentra y nos devuelve la equivalencia en la ontología que deseemos.

Lo que realiza COM, es verificar primero si el concepto se encuentra en la propia ontología, si no es así entonces inicia un proceso de búsqueda de equivalencia entre ambos conceptos, navegando a través de los árboles de conceptos de las ontologías en cuestión.

Cuando ha encontrado una equivalencia, puede aprender el concepto encontrado agregando un nodo o varios al propio árbol. Cabe mencionar que este proceso no es una copia de un pedazo de un árbol a otro, sino un proceso de negociación utilizando equivalencias de conceptos.

CONCLUSIONES

Se ha presentado las características, requerimientos de un modelo de agentes con propósito, los cuales para alcanzarlos deben participar en interacciones ejecutando los papeles para los que cubren sus requisitos.

Se ha indicado la gramática de un lenguaje de interacciones entre agentes con propósitos y la especificación de su interprete.

Los casos de prueba presentados son sobre comercio y se pueden aplicar al comercio electrónico y son susceptibles de mejorarlos.

A partir de los que se ha presentado algunos trabajos que podrían desarrollarse son:

- Un programa que dada una descripción textual en forma lineal genere los diferentes papeles de un ambiente de interacción.
- Un programa que haga la traducción de los programas escritos en LIA al lenguaje Java para aprovechar las características de este último.
- Hacer el interprete distribuido

REFERENCIAS

Amandi Analia, Price Ana, “Towards Object-Oriented Agent Programming: The Brainstorming Meta-Level Architecture”, *Proceedings of Autonomous Agents 97*, Marina del Rey, CA, USA, 1997

Amori Richard D., “An Adversarial Plan Recognition System for Multi-agent Airbone Threats“, Computer Science Department, East Stroudsburg University, 1992

Ayala Gerardo, Yano Yoneo, A Collaborative Learning Environment Based on Intelligent Agents, in Expert Systems with Applications ISSN 0957-4174, vol. 14, Number 1/2 January/February 1998

Baclace Paul E., Competitive Agents for Information Filtering, Communications of the ACM, Dec. 1992, No. 32

Barret Rob, Maglio Paul P., Kellem Daniel C., WBI: A Confederation of Agents that Personalize the Web, in Proceedings of Autonomous Agents 97, Marina del Rey, CA, USA

Bates Joshep, The Role of Emotion in Believable Agents, in Communications of the ACM, July 1994, Vol. 37, No. 7

Boden Margaret A., Agents and Creativity, in Communications of the ACM, July 1994, Vol. 37, No. 7

Canfield Smith David, et. Al., KIDSIM: Programming Agents without a Programming Language, in Communications of the ACM, July 1994, Vol. 37, No. 7

Chavez Anthony, Maes Pattie, Kasbah: An Agent Marketplace for Buying and Selling Goods, MIT Media Lab, Cambridge, MA, USA, 1997

Conrad Stefan, et al., Towards Agent-Oriented Specification of Information Systems, in Proceedings of Autonomous Agents 97, Marina del Rey, CA, USA

Erceau Jean, Ferber Jacques, La Inteligencia Artificial Distribuida, in Mundo Científico 116 May Volume 11

Etzioni Oren, Weld Daniel, A Softbot-Based Interface to the Internet, in Communications of the ACM, July 1994, Vol. 37, No. 7

Finin Tim et. al., Specification of the KQML Agent Communication Language, DARPA Knowledge Sharing Initiative, June 15, 1993

Finin Tim et. al., KQML as an Agent Communication Language, CIKM 94 november 1994, Gaitherburg, MD USA

Grand Stephen, et. al., Creatures: Artificial Life Autonomous Software Agents for Home Entertainment, in Proceedings of Autonomous Agents 97, Marina del Rey, CA, USA

Gray Robert S., Agent Tcl, in Dr. Dobb's Journal, March 1997

Guha R.V., Lenat Douglas B., Enabling Agents to Work Together, in Communications of the ACM, July 1994, Vol. 37, No. 7

Guzmán Adolfo, Finding the Main Themes in a Spanish Document, in Expert Systems with Applications ISSN 0957-4174, vol. 14, Number 1/2 January/February 1998

King William Josseph, Ohya Jun, The Representation of Agents: Antropomorphism, Agency and Intelligence, CHI '96 Companion, Vancouver, BC, Canada

Lester James C., et. Al., The Persona Effect: Affective Impact of Animated Pedagogical Agents, CHI 97, Atlanta, GA, USA

Maes Pattie, Agents that Reduce Work and Information Overload, in Communications of the ACM, July 1994, Vol. 37, No. 7

Maes Pattie, Artificial Life Meets Entertainment: Lifelike Autonomous Agents, in Communications of the ACM, November 1995, Vol. 38, No. 11

Minsky Marvin, The Society of Mind, Simon & Schuster Inc. 1985

Moon Youngme, Nass Clifford, Adaptive Agents and Personality Change: Complementary versus Similarity as Forms of Adaption, CHI '96 Companion, Vancouver, BC, Canada

Noriega Blanco V. Pablo C., Agent Mediated Auctions: The Fishmarket Metaphor, Memory to obtain his Ph.D., Universitat Autònoma de Barcelona, Bellaterra, December 12th, 1997

Riecken Doug, M: An Architecture of Integrated Agents, in Communications of the ACM, July 1994, Vol. 37, No. 7

Rus Daniela, Gray Robert, Kotz David, Transportable Information Agents, in Proceedings of Autonomous Agents 97, Marina del Rey, CA, USA

Sanchez J.Alfredo, Leggett John J., Schnase John L., AGS: Introducing Agents as Services Provided by Digital Libraries, DL 97 Philadelphia PA, USA

Selker Ted, COACH: A Teaching Agent that Learns, in Communications of the ACM, July 1994, Vol. 37, No. 7

Shneiderman Ben, Maes Pattie, Direct Manipulation VS Interface Agents, Interactions November-December 1997

Tu Xiaoyuan, Terzopoulos Demetri, Artificial Fishes: Physics, Locomotion, Perception, Behavior, 1994

Virdhagriswaran Sankar, et. Al., Standardizing Agent Technology, StandardView Vol. 3, No. 3, September 1995

Paton W., Norman, Díaz Oscar, “Active Database Systems”, ACM Computing Survey,Vol, 31, No. 1, March 1999.

Zita Haigh Karen, Veloso Manuela M., High-Level Planning and Low-Level Execution: Towards a Complete Robotic Agent, in Proceedings of Autonomous Agents 97, Marina del Rey, CA, USA