xx (108188.1)

CINVESTAV

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

Metodología para el Análisis y Diseño de Sistemas MultiAgente

Tesis que Presenta
Nicandro Farías Mendoza

Para obtener el grado de
Doctor en Ciencias

En la especialidad de
Ingeniería Eléctrica

Guadalajara, Jal. Noviembre de 2002

Metodología para el Análisis y Diseño de Sistemas MultiAgente

Tesis de Doctorado en Ciencias
Ingeniería Eléctrica

Por:

Nicandro Farías Mendoza

Maestro en Ciencias en Ingeniería Eléctrica
CINVESTAV-IPN

Director de Tesis:

Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Noviembre de 2002

Agradecimientos:

Agradezco a mi esposa:

Lorena Carmina Moreno Jiménez
Por su comprensión
Por su apoyo absoluto.

A mis hijos:

Katia Carmina
Víctor Uriel
Isaac
Por quienes hago este esfuerzo.

Agradecimientos:

A mi asesor:

Dr. Félix F. Ramos Corchado
Por su ayuda, por su confianza, por sus enseñanzas, por su acertada dirección de ésta tesis y por su infinita paciencia y comprensión.

A todos mis profesores:

Por sus enseñanzas, en especial al **Dr. Luis Ernesto López Mellado** por el apoyo que siempre me brindó durante mi estancia en éste centro de estudios.

A los miembros del comité de tesis

Dr. Luis Ernesto López Mellado
Dr. Federico Sandoval Ibarra
Dr. Félix Francisco Ramos Corchado
Dr. Víctor Hugo Zaldívar Carrillo
Dr. Víctor Manuel Larios Rosillo

A la Universidad de Colima:
Por brindarme su patrocinio para realizar mis estudios

Al Instituto Tecnológico de Colima: Por brindarme las facilidades para continuar superándome académicamente

Lista de Figuras

Lista de Tablas

Contenido

Capítulo 1

Introducción

1.1 Definición del problema

En esta sección se presentan en primer lugar, los conceptos con los cuales poder caracterizar el marco contextual del problema a resolver; posteriormente se determina: el problema que se va a resolver, los aspectos que motivaron este trabajo, los objetivos que se proponen alcanzar y finalmente se presenta brevemente la estructura que define el contenido de este trabajo de tesis.

1.1.1 Antecedentes del Problema

El desarrollo de Software hoy en día es una tarea compleja que requiere la consideración de una diversidad de aspectos, los cuales deben tomarse en cuenta para satisfacer las exigencias que demanda la industria, de un Software vanguardista: adecuado a los cambios tecnológicos y ajustado a los nuevos modos de procesamiento de la información. A continuación se hace una breve descripción de los aspectos principales sugeridos para alcanzar una adecuación apropiada en desarrollo del Software en la actualidad.

Los cambios tecnológicos están presentes en: 1) La evolución de las plataformas y modelos de comunicación de información; principalmente las redes de computadoras actuales, las cuales están evolucionando a un ritmo muy acelerado y en donde el tamaño de éstas redes contemporáneas, como la Internet, las Intranets y las redes entre organizaciones; se refleja en un incremento tanto en los dispositivos físicos (impresoras, discos duros, microprocesadores, etc.) como en las fuentes de información que las constituyen. Como ejemplo de estas fuentes de información, citamos las siguientes: Bases de Datos, Páginas WEB, Direcciones URL, Agentes de aplicación etc. Todos estos recursos antes escasos se encuentran distribuidos a través de las redes mencionadas. 2) Los paradigmas empleados para el desarrollo de Software, los cuales deben adecuarse tanto a los cambios en la tecnología como a las necesidades que pretenda la industria.

Los nuevos modos de procesamiento de la información son advertidos tomando en cuenta las siguientes observaciones: 1) Los usuarios de los sistemas de Software demandan aplicaciones que resuelvan sus necesidades de un modo: a) *simple*; es decir, fácil de utilizar; b) *transparente*: esto es, al usuario no le importa en donde se encuentren las fuentes de información ni los medios empleados para manipularla, el usuario solo requiere cierta información específica; c) *confiable*: el usuario requiere que el resultado de las transacciones que realiza, sean confiables y seguras. 2) Asimismo los usuarios demandan que las aplicaciones que realizan sean presentadas en un contexto virtual: es decir los usuarios solicitan la ejecución de transacciones bancarias; pago de servicios diversos; accesos a fuentes

de información; compra y venta de artículos, etc. Todo en forma electrónica, desde su casa, o desde su oficina.

De lo expuesto previamente, se deduce que en desarrollo de los sistemas de Software en la actualidad requiere de entidades autónomas dispersas en ambientes distribuidos, que actúen de manera inteligente para resolver un problema especifico de manera amigable al usuario, de tal manera que la solución obtenida sea coherente, cooperativa, segura y amigable al usuario.

Por otra parte debido al incremento exponencial de los recursos de Software y de Hardware dispersos en la Internet. La automatización de la búsqueda de fuentes de información, la selección y búsqueda de agentes relevantes, la selección y control de los dispositivos interconectados en la red, es esencial por varias razones:

Primero: Los usuarios en el Cyber espacio pueden no tener idea de en donde encontrar un servicio y que agentes están disponibles para ejecutar una tarea. *Segundo*: Es posible que los usuarios no se den cuenta de los cambios que ocurren en la Internet, es decir, las fuentes de información y los agentes relevantes pueden aparecer y desaparecer en cualquier instante de tiempo. Asimismo los dispositivos físicos pueden conectarse y desconectarse en el tiempo. *Tercero*. Como la cantidad de recursos y la sofisticación de los mismos se incrementa, existe la necesidad de una estandarización, tanto en los mecanismos de comunicación entre los agentes como en los mecanismos de coordinación con los cuales los agentes puedan ejecutar una tarea en forma cooperativa, coherente y segura.

Este crecimiento de los recursos de información residentes en la red requiere de sistemas de información que puedan estar distribuidos en la red y en interacción con otros sistemas. Estos sistemas no pueden realizarse con las tecnologías de Software Tradicionales, debido a las limitaciones de estas tecnologías para manejar la distribución e interoperabilidad.

La Tecnología basada en agentes ha mostrado ser útil para facilitar la realización de sistemas de información distribuidos e interoperables, debido a que esta fue creada para hacer frente a los problemas de la distribución e interacción mutua [28].

1.1.2 El Problema

Hoy en día la industria manifiesta una demanda para el desarrollo de aplicaciones soportadas por redes de computadoras, en donde estas redes pueden ser: Intranets, redes corporativas la Internet o el Web. Como se observa en la Fig. 1.1, la construcción de estas aplicaciones requiere: 1) la descripción de un conjunto de componentes capaces de percibir y actuar de manera autónoma y colaborativa en el medio ambiente en el que están situados, a estos componentes les llamamos agentes; 2) definir para estos agentes una estructura con la cual construir una organización que represente una solución posible del problema a resolver; 3) definir el lenguaje por medio del cual los agentes logren una interacción mutua; 4) Describir las reglas y políticas para que los componentes y organizaciones de componentes puedan alcanzar una solución coordinada, conjunta y negociada; 5) también se requiere de procedimientos de seguridad integridad y coherencia, con los cuales, alcanzar la calidad necesaria en la solución de esta clase de problemas distribuidos.

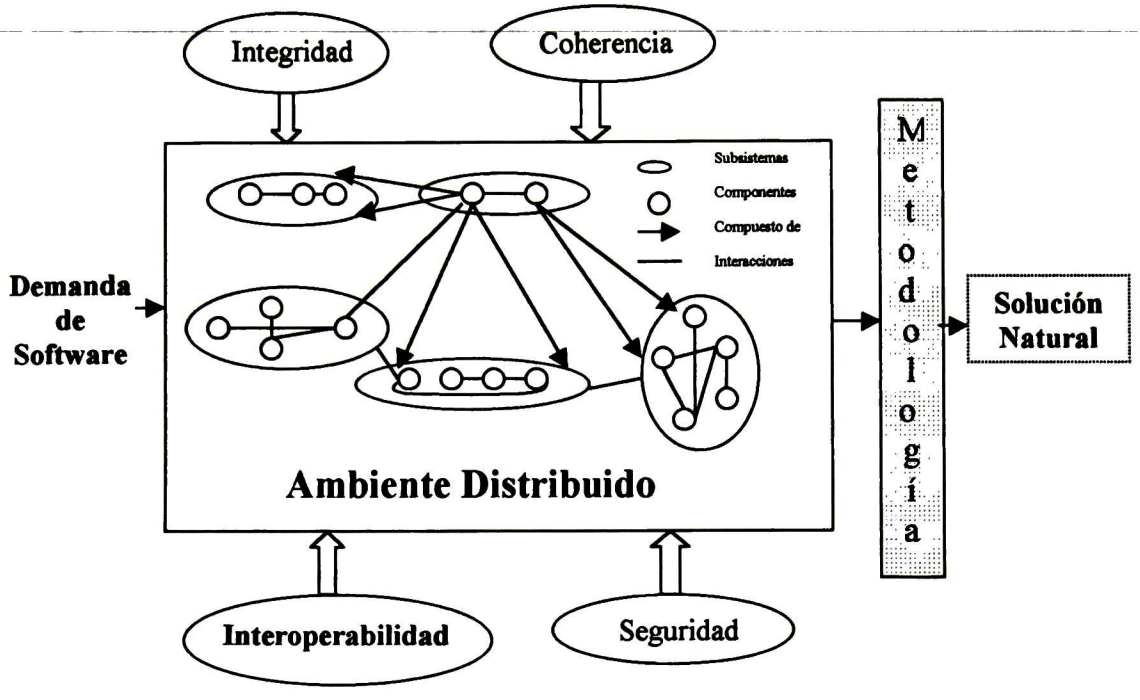


Fig. 1.1 Contexto del problema

Como se describió anteriormente la solución de los problemas complejos que demanda actualmente la industria, requiere de un nuevo enfoque que englobe los principios y conceptos con los cuales obtener de manera sistemática y ordenada una solución obtenida en forma natural a los problemas enmarcados en ambientes distribuidos. Es decir, para hacer frente a los problemas que demanda la industria es necesario contar con una metodología con la cual sea posible obtener una solución simple y amigable a esta clase de problemas.

El problema detectado, consiste en la carencia de una metodología, adecuada para implantar los problemas que deben resolverse para desarrollar el tipo de sistemas que demanda la industria actualmente, por ejemplo aplicaciones de comercio electrónico, control de aeropuertos, transacciones bancarias, etc.

Este problema de establecer una metodología es derivado, como lo expusimos anteriormente: *Primero*, por la dificultad para poder abstraer la secuencia de pasos para resolver de manera disciplinada el problema que involucra a un conjunto de entidades en interacción, distribuidos en la red, con el propósito de resolver un problema en forma cooperativa. *Segundo*, por la carencia de herramientas tecnológicas para hacer frente a los problemas de abstracción e interacción de los sistemas distribuidos en la red. *Tercero*: por la dificultad para proporcionar la seguridad apropiada en sistemas de seguridad crítica como administrar la gran cantidad de recursos inmersos en la Internet. *Cuarto*: Por los cambios dados en los modos de procesamiento hoy en día, es decir, actualmente se tiene una tendencia hacia en procesamiento electrónico de la información, con las consecuencias que este tipo de procesamiento acarrea.

Tomando el paradigma de agente el problema se resume en la dificultad de poder abstraer:

- Un modelo que describa la estructura completa de un Sistema MultiAgente.
- Un modelo que especifique de manera clara y precisa las diversas clases de agentes y las relaciones que se presentan entre estas clases.
- Un sistema formal con el cual sea posible especificar y verificar un sistema MultiAgente.
- Un prototipo con el cual sea posible capturar la funcionalidad individual de los agentes.
- Un modelo que gobierne el intercambio de mensajes entre los agentes, es decir un modelo que exhiba el comportamiento dinámico de los agentes.
- Un sistema MultiAgente enfocado a las operaciones de comercio electrónico.

1.2 Motivación

A continuación se mencionan los principales aspectos que motivaron el desarrollo de este trabajo de tesis:

- En un futuro se vislumbran más y más sistemas cooperativos, en donde se tendrán aplicaciones de agentes que recorran las redes de información, realizando tareas específicas:
  - Sistemas basados en conocimiento
  - Sistemas Cooperativos
  - Sistemas de Planeación
  - Sistemas Abiertos
- Las industrias, cuyas aplicaciones emplean las redes necesita de una tecnología versátil para enfrentar los retos que se le presentan:
  - Migración de los poco apropiados sistemas centralizados hacia los sistemas de operación distribuida.
  - Adecuación a los nuevos modos de procesamiento de la información:
    - Negocio Electrónico
    - Comercio Electrónico
    - Realidad virtual
- Las deficiencias en las herramientas de desarrollo actuales: La tecnología de la Internet y el WEB proporcionan ambientes realmente distribuidos. Estos ambientes son la base para el desarrollo de Sistemas MultiAgente, sin embargo, no es suficiente contar con una plataforma que solo ofrece el medio de transmisión y el acceso a las aplicaciones actuales. La Internet no ofrece los mecanismos suficientes para asegurar la validez, seguridad e integridad e la información procesada dentro de su contexto.

1.3 Objetivos

El Objetivo general de la tesis consiste en desarrollar una metodología que facilite el desarrollo de Sistemas MultiAgente. Esta metodología debe abarcar los modelos conceptuales, sean estos abstractos o concretos, de tal manera que nos proporcionen un modelo estático, un modelo funcional y un modelo dinámico del sistema que se está desarrollando.
Los objetivos específicos de este trabajo de tesis son:

- Precisar la estructura general para la metodología propuesta
- Definir un modelo estático para el SMA que se va a desarrollar. Este modelo involucra a los agentes que intervienen en el sistema y especifica la forma en que estos agentes están relacionados.
- Especificar el Metalenguaje formal para modelar el SMA que se está desarrollando. Este Metalenguaje está basado en la lógica temporal de creencias.
- Definir el lenguaje Objeto. Basado en el paradigma "Capacidad de Interacción"
- Exponer la Arquitectura individual del agente. La cual está basada en el modelo de Arquitectura deliberativa.
- Especificación de los protocolos de interacción: Coordinación, Cooperación y Negociación.
- Describir la Aplicación, orientada a operaciones de comercio electrónico.

En este trabajo de tesis se hace la propuesta de una metodología, con la cual hacer frente a los problemas del desarrollo de Software que demanda la industria en la actualidad. La Fig. 1.2 representa un esquema en el cual se muestran los principales conceptos de ésta metodología.

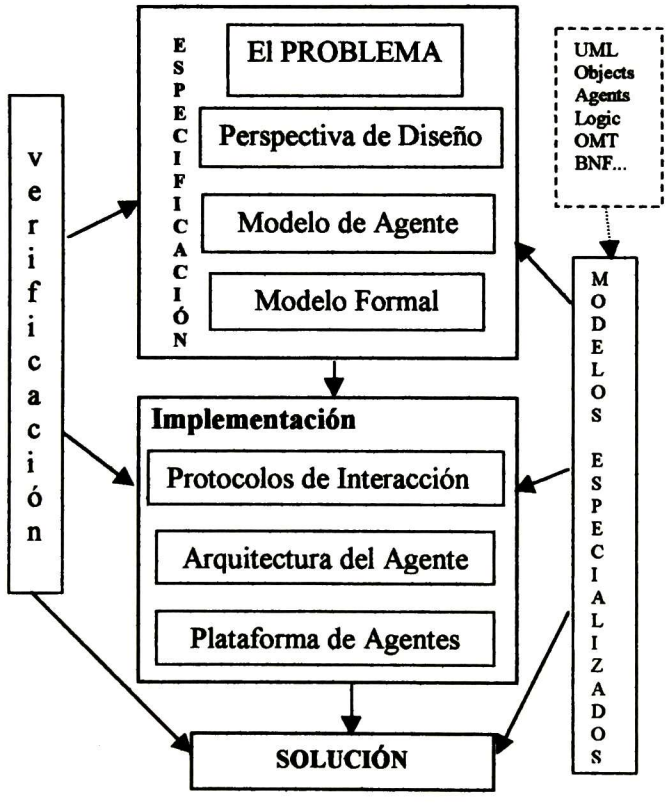


Fig. 1.2 Esquema de la Metodología propuesta

Los modelos conceptos y procedimientos incorporados en la Metodología propuesta son agrupados básicamente en tres fases: 1) Una fase de especificación, en donde se precisa el problema a resolver y se describen las propiedades que es sistema debe exhibir. 2) una fase de

implementación, en la cual se traslada la especificación del problema a resolver en un sistema de cómputo concreto. 3) Una fase de verificación cuyo propósito es el de garantizar que se está desarrollando correctamente la solución al problema propuesto.

La metodología toma como entrada el problema a resolver y utiliza un conjunto de modelos especializados, con los cuales poder elaborar un procedimiento sistemático que nos conduzca a una solución apropiada del problema que se está resolviendo. A continuación se hace una breve descripción de algunos de los componentes mostrados en la Fig. 1.2.

Perspectiva de diseño: Como entrada la perspectiva de diseño recibe la descripción del problema que se debe solucionar. Además de un conjunto de modelos especializados. Estos modelos especializados proporcionan el soporte que nos permite precisar los aspectos relevantes del SMA a resolver.

modelo de agente: Este modelo captura la estructura estática del sistema, mostrando los agentes involucrados en el sistema, las relaciones existentes entre ellos y los atributos que caracterizan a cada clase.

Modelo formal: Haciendo referencia al modelo de agentes, se construye un modelo basado en la lógica temporal lineal. Con este modelo se especifica formalmente el SMA. Este modelo formal utiliza el Metalenguaje (AML) para dar una interpretación semántica a las acciones de los agentes. AML tiene asociado al lenguaje objeto LCIASA [8]. Con LCIASA se da una interpretación computacional al modelo formal. Es decir LCIASA proporciona la estructura sintáctica y semántica del SMA especificado.

Arquitectura del agente: La arquitectura del agente describe la funcionalidad del SMA. Es decir, aquí se describe como ocurren los cálculos dentro del sistema.

Protocolos de Interacción: La interacción de agentes abarca los aspectos del sistema que están relacionados con el intercambio de información, las políticas de comunicación entre los agentes y los protocolos utilizados para la conversación entre los agentes.

Plataforma de agentes: Define el marco de trabajo que nos permite organizar y realizar operaciones con los agentes con el propósito de obtener una solución concreta del problema específico a resolver.

La metodología propuesta ofrece las siguientes ventajas sobre otros modelos basados en el paradigma de agentes:

- Engloba los modelos que describen de manera precisa la estructura estática, funcional y dinámica de un SMA.
- Tiene asociado Metalenguaje Formal con el cual es posible verificar las acciones contempladas por un SMA en la solución de un problema específico.
- Cuenta con un Lenguaje Objeto llamado LCIASA[5], el cual está basado en el concepto del speech Act. Con este lenguaje se expresan de manera clara, las acciones que los agentes llevan a cabo en la solución de un problema determinado.

- Con MSMA es posible garantizar la validez, seguridad y coherencia de los SMA implantados bajo esta metodología.

MSMA se presenta como una herramienta apropiada para desarrollar aplicaciones en ambientes distribuidos abiertos, asimismo con MSMA es posible adaptarse a los nuevos modos de procesamiento de la información: Negocio Electrónico, Comercio Electrónico, ambientes virtuales, etc.

1.4 Estructura de la tesis

La tesis esta organizada en siete capítulos. En estos capítulos se muestra un panorama que exhibe una idea completa de la metodología propuesta para el desarrollo de SMA. Esta metodología abarca desde la estructura general del SMA, la perspectiva de diseño, la arquitectura del agente, los protocolos de interacción, la aplicación, hasta la plataforma de desarrollo del sistema.

El capítulo 2 expone el estado del arte de los sistemas y modelos basados en el paradigma MultiAgente. Aquí se dan las definiciones y conceptos que muestran la evolución de los paradigmas que representan los antecedentes de la metodología propuesta.

El capítulo 3 presenta la metodología propuesta para el Análisis y Diseño de Sistemas MultiAgente. La presentación emplea una arquitectura de capas, para describir los componentes incluidos en la metodología. Asimismo se hace una breve descripción de cada uno de estos componentes.

El capítulo 4 muestra la fase de especificación en la cual se obtiene primero un modelo conceptual del SMA y segundo se obtiene un modelo formal para especificar y modelar las acciones que los agentes ejecutan en la consecución de una aplicación. Este modelo formal abarca un modelo de agente y un modelo formal, basado en la lógica temporal. Tercero, se describe el lenguaje objeto LCIASA, utilizado para definir una estructura sintáctica y semántica para la secuencia de mensajes intercambiados por los agentes y finalmente se presentan los protocolos de interacción, con los cuales de guía el intercambio de mensajes entre los agentes. Los protocolos de interacción considerados en esta metodología son: Coordinación, Cooperación y Negociación.

El capítulo 5 exhibe la fase de implementación, en donde primero se propone la arquitectura del agente, la cual representa el punto de vista interno en la especificación del agente. Esta arquitectura contiene un modelo simbólico del mundo en donde las decisiones son hechas por medio del razonamiento lógico. Segundo se hace propone el protocolo "Contract Business", (CB), el cual es implementado utilizando el lenguaje LCIASA, para establecer la solución del problema de comercio electrónico propuesto como caso de estudio.

El Capítulo 6 Muestra la fase de verificación, en donde se proponen los siguientes mecanismos de verificación: Verificación del ciclo de vida y verificación formal.

El capítulo 7 presenta las conclusiones de este trabajo de tesis, asimismo se presentan y se hacen las recomendaciones que guían los posibles trabajos futuros de esta disertación.

Capítulo 2

Estado del Arte

Resumen

En éste capítulo se expone de manera breve conceptos y fundamentos que caracterizan a los distintos paradigmas incorporados a la Ingeniería de Software para el desarrollo de sistemas de Software.

2.1 Introducción

El diseño y construcción de Software con eficacia industrial y de alta calidad es una tarea difícil. Con el fin de hacer que el proceso para desarrollo de sistemas de información sea más sencillo, se ha concebido un amplio rango de paradigmas para el desarrollo de Software, de tal manera que sea posible construir las aplicaciones complejas. En la actualidad los paradigmas que tienen un mayor grado de aceptación, son las que están basadas en diagramas de flujo de datos [51]. Aunque los enfoques Orientados a Objetos (OO) [5], cada vez están ganando mayor terreno en el campo del desarrollo de software. Sin embargo, debido a las razones expuestas anteriormente, el enfoque de desarrollo Orientado Agentes (OA) [28] está comenzando a aparecer como una propuesta prometedora para el desarrollo de aplicaciones complejas.

2.2 Paradigmas para el desarrollo de Software

A continuación citamos y describimos brevemente los paradigmas de mayor importancia utilizados para el desarrollo de Software, haciendo una breve descripción de cada uno de ellos:

- Análisis Estructurado/Diseño Estructurado (SA/SD)
- Desarrollo estructurado de Jackson (JSD)
- Desarrollo de Sistemas Estructurados de Datos (SDSD)
- El Análisis Estructurado y la Técnica de Diseño (SADT)
- Técnica de modelado de Objetos (OMT)
- El método de Boosh
- El método de Jacobsen (Objectory)
- El Lenguaje Unificado de Modelado (UML)
- Orientación Agentes (AO)

2.2.1 Análisis Estructurado/Diseño Estructurado (SA/SD)

El Análisis Estructurado/Diseño estructurado admite los tres modelos ortogonales: modelo de objetos, modelo funcional y el modelo dinámico, ésta metodología hace énfasis en la descomposición funcional. El sistema se ve sobre todo como algo que proporciona una o más funciones al usuario final. SA/SD está en todas partes, está bien documentado y se aplica a la solución de muchos problemas reales [56].

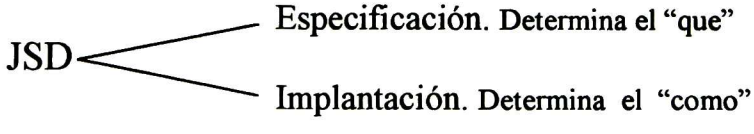
SA/SD incluye toda una gama de notaciones para especificar formalmente el software. Durante la fase de análisis, se utilizan diagramas de flujo de datos, especificaciones de procesos, un diccionario de datos, diagramas de transición de estados y diagramas de Entidad Relación para describir lógicamente el sistema. En la fase de Diseño se añaden detalles a los modelos de la fase de análisis y se transforman los diagramas de flujo de datos en descripciones de cartas de estructuras del código del lenguaje de programación.

Los estudios sobre SA/SD se han realizado principalmente por Coad y Yourdon [56]. Se considera con frecuencia a SA/SD como uno de los métodos de Análisis orientado a objetos más sencillos de aprender. El método de Coad y Yourdon es uno de los métodos de AOO que presentan una mayor facilidad para su estudio. La notación del modelado es relativamente simple y las reglas para desarrollar el modelado del análisis son evidentes. A continuación se presenta una descripción resumida del proceso de AOO de Coad y Yourdon.

- Identificar los objetos usando el criterio de "que buscar"
- Definir una estructura de generalización – especificación.
- Definir una estructura de Todo – parte.
- Identificar temas (representación de subsistemas).
- Definir atributos.
- Definir servicios.

2.2.2 Desarrollo estructurado de Jackson (JSD)

Esta metodología es popular en Europa, no distingue entre Análisis y Diseño y junta ambas fases en una Especificación [25].



JSD utiliza modelos gráficos durante el desarrollo del sistema, está destinada a las aplicaciones en las cuales la temporización es un factor importante. Los modelos JSD describen el mundo en términos de entidades, acciones y secuencias de acciones. Las

entidades aparecen como sustantivos en las exposiciones de requisitos y las acciones aparecen como verbos. El desarrollo de Software JSD consiste de los siguientes seis pasos secuenciales:

1. Entidades y Acciones. Se identifican las entidades: personas objetos u organizaciones que un sistema ocupa para producir o utilizar información y se identifican las acciones: los sucesos que ocurren en el mundo real que afectan a las entidades.
2. Estructura de entidades: Acciones que afectan a cada entidad se ordenan por tiempo y se representan como diagramas de Jackson (notación tipo árbol).
3. Modelado inicial: Las entidades y las acciones se representan como un modelo de proceso, se definen las conexiones entre el modelo y el mundo real.
4. Paso de función: utiliza un pseudocódigo para indicar las salidas de las acciones.
5. Planificación del sistema: se evalúan y se especifican las características de planificación de procesos.
6. Implementación: Se centra en los problemas de planificación de procesos y asigna procesadores a los procesos.

2.2.3 Desarrollo de Sistemas Estructurados de Datos (SDSD)

El Desarrollo de Sistemas Estructurados de Datos (SDSD) también conocido como Metodología de Warnier-Orr, evolucionó del trabajo de J.D. Warnier [52], quien desarrolló una notación para representar la Jerarquía de información mediante tres construcciones: Por secuencia, selección y repetición y demostró que la estructura de Software se podría derivar directamente de la estructura de datos. Kerr Orr Amplió el trabajo de Warnier para acompañar de una visión algo más extensa del dominio de información en el desarrollo de los sistemas estructurados de datos. El SDSD estudia el flujo de información y características funcionales, así como la jerarquía de datos.

2.2.4 Análisis Estructurado y la Técnica de Diseño (SADT)

El Análisis Estructurado y la Técnica de Diseño (SADT). Es una técnica que se ha utilizado mucho como una notación para la definición de sistemas, representación de procesos, Análisis de requisitos de Software y diseño de sistemas/software [43]. SADT se compone de procedimientos que permiten al analista, descomponer las funciones del Software (o sistema); una notación gráfica, el actigrama y datagrama SADT que comunica las relaciones de información (datos y control) y función dentro del Software y las directrices de control de proyectos para aplicar la metodología.

La metodología SADT es acompañada de herramientas automatizadas para poder soportar procedimientos de análisis y disposiciones de organización para poder utilizar correctamente las herramientas. Se especifican las revisiones y los hitos, permitiendo la validación de la comunicación cliente - servidor.

2.2.5 Técnica de Modelado de Objetos (OMT)

La Técnica de modelado de Objetos (OMT) fue desarrollada principalmente por J. Rumbaugh [45]. La metodología OMT está basada en el desarrollo de un modelo del sistema

con tres partes, que después se refinan y optimizan para construir un diseño. El modelo de objetos captura los objetos del sistema y sus relaciones, el modelo dinámico describe la reacción de los objetos del sistema frente a sucesos, también las interacciones entre objetos. El modelo funcional especifica las transformaciones de objetos y las restricciones aplicables a estas transformaciones. A continuación se muestra una breve descripción del proceso de AOO de Rumbaugh.

- Desarrollar una declaración de ámbito del problema.
- *Desarrollar un modelo de objetos*
  Identificar clases relevantes del problema
  Definir atributos y asociaciones.
  Definir enlaces de objetos.
  Organizar las clases de objetos usando la herencia.
- *Desarrollar un modelo dinámico*
  Preparar escenarios
  Definir eventos y desarrollar una taza de eventos para cada escenario.
  Construir un diagrama de flujo de eventos.
  Desarrollar un diagrama de estados
  Revisar el comportamiento para revisar la consistencia y realización.
- *Desarrollar un modelo funcional para el sistema.*
  Identificar entradas y salidas
  Usar diagramas de flujo para representar información del flujo.
  Desarrollar EP (especificación de procesos) para cada función.
  Especificar criterios de restricciones y optimización.

2.2.6 Método de Booch.

El método de Booch abarca un "microproceso de desarrollo" y un macroproceso de desarrollo. El nivel micro define un conjunto de tareas de análisis que se reaplican en cada etapa en el macroporceso. El método de Booch está soportado por una variedad de herramientas especializadas. A continuación esbozamos brevemente el microproceso de desarrollo del AOO de Booch [5].

- Identificar clases y Objetos.
  Proponer objetos candidatos.
  Conducir el análisis de comportamiento.
  Identificar escenarios relevantes.
  Definir atributos y operaciones para cada clase.
- Identificar la semántica de clases y objetos.
  Seleccionar y analizar escenarios.
  Asignar responsabilidades para alcanzar el comportamiento deseado.
  Dividir las responsabilidades para equilibrar el comportamiento.
  Seleccionar un objeto y papeles y responsabilidades.
  Buscar colaboración con otros objetos.
- Identificar relaciones entre clases y objetos.
  Definir las dependencias que existen entre objetos.

Describir el papel (rol) de cada objeto participante.
Validar los escenarios por revisión completa.

- Realizar una serie de refinamientos.
  Producir diagramas apropiados para el trabajo realizado en los puntos anteriores.
  Definir Jerarquías de clases apropiadas.
  Crear agrupamientos basados en clases comunes.
- Implementar clases y objetos.

2.2.7 Método de Jacobsen (Objectory)

Este método también es llamado ISOO (Ingeniería de Software Orientada a Objetos), el Método de Jacobsen, es una versión simplificada de Objectory, método desarrollado por el mismo Jacobnsen [26]. Este método se diferencia de otros enfoques OO por la importancia que da al *caso de uso*: Una descripción o escenario que describe como el usuario logra tener una acción interactiva con el producto o sistema. A continuación se hace un breve bosquejo del proceso de AOO de Jacobsen:

- Identificar los usuarios del sistema y los usuarios globales.
- Construir un modelo de requisitos.
  Definir los Actores y sus responsabilidades.
  Identificar los casos de uso para cada actor.
  Preparar una visión inicial de los objetos del sistema y sus relaciones.
  Revisar el modelo usando los casos de uso como escenarios para determinar su validez.
- Construir un modelo de análisis.
  Identificar objetos de interfaz usando información del tipo actor - interacción.
  Crear vistas estructurales de los objetos de interfaz.
  Representar el comportamiento del objeto.
  Aislar subsistemas y modelos.
  Revisar el modelo usando casos de uso como escenarios para determinar su validez.

2.2.8 El Lenguaje Unificado de modelado (UML)

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de Software. Captura conocimiento y decisiones sobre los sistemas que se deben construir. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado. UML combina principalmente los métodos de Runbaugh (OMT), Booch y Jacobson (Objectory) e incorpora las mejorares prácticas actuales en un acercamiento estándar. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. El surgir de UML parece ser atractivo a la generalidad del público informático porque consolida la experiencia de varios autores con un estado oficial que reducirá las diferencias gratuitas entre herramientas [44].

En la descripción de un sistema UML organiza conceptos de alto nivel en un pequeño conjunto de diagramas y vistas que presentan los conceptos en forma visual. La división en

diversas vistas es algo arbitraria, tratando que esta división sea los más intuitiva posible. Una o dos clases de diagramas proporcionan una notación visual para cada vista.

A continuación se presentan los conceptos de UML para la descripción de un sistema

- Clasificación estructural
  - ➢ Vista estática
    Diagrama de clases
  - ➢ Vista de casos de uso
    Diagrama de casos de uso
  - ➢ Vista de implementación
    Diagrama de componentes
  - ➢ Vista de despliegue
    Diagrama de despliegue

- Comportamiento dinámico
  - ➢ Vista de máquina de estados
    Diagrama de estados
  - ➢ Vista de actividad
    Diagrama de actividad
  - ➢ Vista de interacción
    Diagrama de secuencia
    Diagrama de colaboración

- Gestión del modelo
  - ➢ Vista de gestión del modelo
    Diagrama de clases

UML pretende trabajar correctamente con todos o al menos con la mayoría de los procesos de desarrollo existentes. Un objetivo final de UML es el de ser tan simple como sea posible pero manteniendo la capacidad de modelar toda gama de sistemas que se requiera construir.

La técnica de modelado de objetos produce sistemas que son más estables, con respecto a los cambios de requisitos que las aproximaciones tradicionales orientadas a funciones.

Estos paradigmas han sido aplicados ampliamente para el diseño y construcción de sistemas de Software. Sin embargo, tanto los distintos enfoques del paradigma Orientado al análisis estructurado como los distintos enfoques del paradigma orientado a Objetos, no cumplen con las expectativas requeridas para obtener una solución aceptable de los problemas complejos que demanda la industria del Software en la actualidad.

2.2.9 Orientación Agentes (OA)

El desarrollo de Software Orientado Agente representa un nuevo enfoque para abstraer y conceptuar la solución de problemas distribuidos y colaborativos. Este enfoque está basado en los siguientes conceptos básicos [28]:

i). El concepto de *granularidad* en el cual un problema se divide en un número de componentes autónomos los cuales pueden ser operados de manera independiente. ii) el concepto de *organización* que se refiere a la agrupación de componentes que se pueden reunir en jerarquías de control o en grupos de componentes del mismo tipo y iii) el concepto de *interoperabilidad* con el cual podemos manejar las interacciones entre los agentes en contextos distribuidos abiertos como la Internet y el WEB.

La aproximación de Análisis, diseño e implementación de Software orientado a Agente, se tiene una visión del mundo en función de una colección de agentes autónomos en interacción, esta aproximación considera las siguientes fases: Especificación, Implantación y Verificación.

En la fase de Especificación se obtienen: los requerimientos del sistema, se definen las propiedades que el sistema es capaz de representar y se define el sistema a resolver. En esta fase se agrupan los modelos: Modelo de agentes, Modelo Formal y Los protocolos de Interacción, Para cumplir con las funciones mencionadas anteriormente. La fase de especificación se aborda con mayor amplitud en el capítulo 4.

En la fase de implementación se estudia el cambio de la especificación abstracta a un sistema computacional concreto. Esta fase agrupa la arquitectura del agente y la plataforma de agentes, esta fase se describe en el capítulo 5.

Finalmente en la fase de verificación se utilizan dos mecanismos de verificación: verificación de ciclo de vida y verificación formal. Esta fase de describe con mayor detalle en el capítulo 6.

Los trabajos más significativos en el enfoque Orientado Agente fueron realizados por:

- La Fundación para Agentes Físicos Inteligentes (FIPA) [17].
- G.M.P.O'Hare and N.R. Jennings. En su texto Foundations on Distributed Artificial intelligence presenta un punto de partida prometedor para la ingeniería de Software [21].
- G. Weiss en su texto: Multi-agent Systems, exhibe las bases del enfoque Orientado Agente[53].
- Ferber Jacques. En su texto Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, hace una descripción amplia de la aplicación del paradigma orientado agente a los sistemas distribuidos [16].
- Nicholas Jennings, Michael Wooldridge. Proponen una metodología para el análisis y diseño de Software Orientado Agente [28].
- Patie Maes. Es una pionera el saber de los Agentes [31].
- Katia Sycara. Ha realizado diversos estudios relacionados con los Agentes [48].
- David Kinny, Michael Georgeff. Proponen una técnica para el modelado y diseño de Sistemas MultiAgente [29].

El paradigma Orientado Agente Cuenta los principios y conceptos para hacer frente de una manera apropiada a los problemas de complejidad que demandan las aplicaciones actuales. Este paradigma constituye la parte fundamental para la construcción de la metodología propuesta, esta metodología es discutida con mayor detalle en el capítulo 3.

Conclusión

En la actualidad se siguen aplicando la mayoría de los paradigmas antes mencionados, no obstante, el paradigma orientado a objetos, en particular la metodología de Booch y UML cada día tienen mayor aceptación para el desarrollo de Software. No obstante el enfoque para el desarrollo de Software Orientado a Objetos tiene serias deficiencias, debido a la pasividad de los objetos y a la falta de conceptos y mecanismos para enfrentar de manera adecuada las aplicaciones extensas, desarrolladas en contextos abiertos y competitivos.

Sin embargo la demanda por parte de la industria para desarrollar Software de calidad, requiere producir aplicaciones caracterizadas por un gran número de componentes en interacción, situados en contextos distribuidos abiertos. Para instrumentar de manera adecuada las aplicaciones demandas por la industria, se requiere de mecanismos y modelos que nos permitan, conseguir los siguientes aspectos: primero; garantizar la seguridad, integridad y coherencia de las aplicaciones elaboradas. Segundo definir las propiedades, la interacción y la organización de los componentes involucrados en la solución de un problema específico.

Estudios recientes han probado que el paradigma Orientado Agente se presenta como una opción prometedora para poder enfrentar los retos derivados de los avances en la tecnología y de los nuevos procedimiento empleados en el desarrollo de las aplicaciones que la industria pretende.

Hoy en día, el paradigma Orientado Agente tiene poca incidencia en el desarrollo de aplicaciones, sin embargo cada día adquiere una mayor aceptación. Se estima que en un plazo corto el uso de éste paradigma se convierta en una necesidad imperativa para el desarrollo de aplicaciones en todos los contextos sociales.

Capítulo 3

Metodología para el Análisis y Diseño de Sistemas MultiAgente

Resumen

Este capítulo presenta los trabajos de investigación que originaron la metodología que se presenta en esta tesis; asimismo se presentan los principios para establecer y explicar los conceptos utilizados para definir la metodología, como el proceso sistemático que abarca un conjunto de etapas, modelos y procedimientos para representar la solución basada en el paradigma de agentes para un problema específico.

3.1 Introducción

La metodología para el análisis y diseño de sistemas MultiAgente (MSMA) es un procedimiento sistemático, el cual toma como concepto fundamental al agente. Con este concepto se crea un conjunto de modelos orientados al paradigma de agente, con los cuales es posible conceptuar, abstraer y resolver los problemas complejos reales que están presentes hoy en día, ya sea que surjan por su importancia, por su naturaleza o por la demanda que tienen por parte de la industria para su implantación.

La metodología propuesta representa un nuevo enfoque interesante para la especificación, implementación y verificación de sistemas de software extensos. Asimismo esta metodología permite mejorar las prácticas para el desarrollo de los sistemas software que se requieren en la actualidad y de esa manera extender el rango de las aplicaciones factibles de ser implementadas.

La importancia de esta metodología, radica en su potencial para enfrentar los problemas de la Distribución e interoperabilidad entre los agentes y de esta forma solucionar problemas en ambientes distribuidos abiertos en forma cooperativa.

3.1.1 Antecedentes

La metodología propuesta tomo como antecedente principal el trabajo de tesis de N. Farías [8], en donde se desarrolla el lenguaje LCIASA, este lenguaje hace un estudio de los Sistemas MultiAgente basado en la capacidad de interacción de los agentes involucrados en la solución de un problema específico. El trabajo realizado para LCIASA fue complementado con los trabajos de investigación realizados por: D. Kinny & M. Georgeff [29]; N. Jennings & M.

Wooldidge [28], K. Sycara [48], P.Maes [31] y por los conceptos y procedimientos definidos en otras metodologias para el desarrollo de sistemas de información [52], [45], [5], [25]. Para evolucionar en la metodología para el análisis y diseño de Sistemas MultiAgente que se propone es este trabajo de tesis.

La fig. 3.1 ilustra de manera gráfica los elementos principales que constituyen el antecedente de la metodología propuesta. En la figura se cita el trabajo de tesis de Z. Aguirre [1], en cual realiza una codificación de la aplicación propuesta en la metodología.

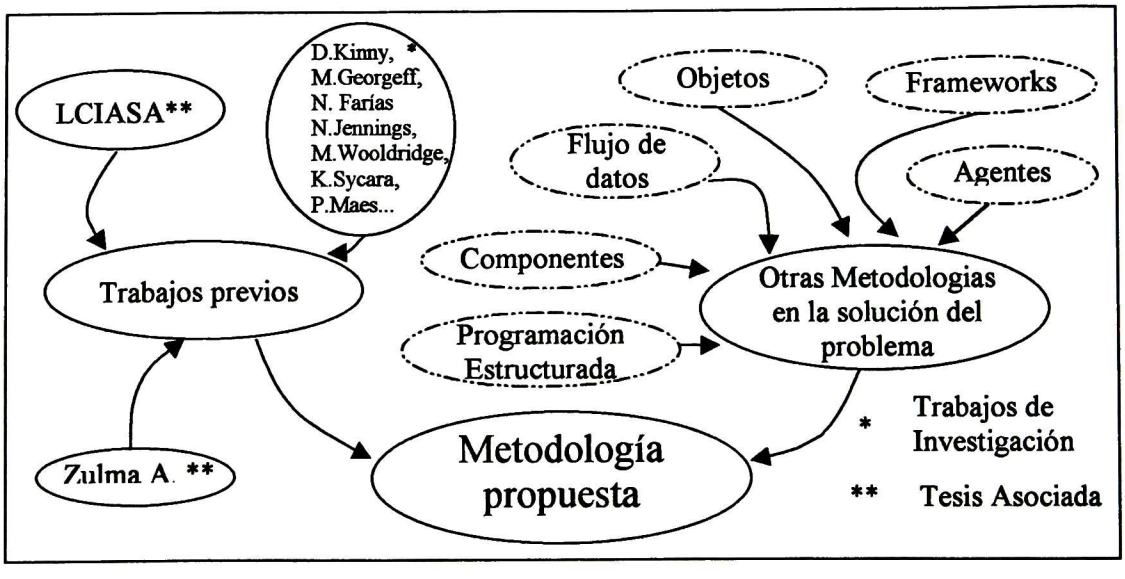


Fig. 3.1 Antecedentes de la metodología propuesta

En esta tesis se presenta una metodología orientada a agentes para el desarrollo de Sistemas MultiAgente. Esta metodología es llamada: Metodología para Sistemas MultiAgente (MSMA), la cual abarca los aspectos que van desde la especificación hasta la verificación, pasando por la implementación de sistemas basados en el paradigma de agente.

3.2 El paradigma de Agente

El concepto de paradigma es considerado como un modelo de pensamiento que posee un cierto número de conceptos y propiedades que lo definen, para así de esta forma, poder dar una visión distinta de la realidad, es decir es posible percibir el mismo contexto de la realidad utilizando los diferentes enfoques obtenidos por distintos paradigmas.
El paradigma de agente nos permite percibir el mundo o la realidad por medio de entidades autónomas, situadas en un medio ambiente, capaces de percibir y actuar en su medio ambiente, las cuales tienen un conjunto de metas y motivaciones que tratan de alcanzar por medio de sus acciones, a estas entidades las llamamos *agentes*. En otras palabras, el paradigma de agente es un modelo basado en el concepto de agente, el cual esta definido por un conjunto

de propiedades y principios que lo caracterizan, y de esta forma poder abstraer y describir el mundo y proporcionar una visión particular de este mundo, por medio de uno o mas agentes. Para poder estimar la clase de problemas que son factibles de representar por medio del paradigma de agentes, a continuación se presentan los conceptos y las propiedades asociadas al paradigma de agente.

3.2.1 Propiedades de los agentes

A continuación se presentan las propiedades más importantes con las cuales se caracteriza al agente, definiendo su potencial y así estimar la clase de los problemas que son posibles de resolver por medio de este paradigma.

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

Reactividad. Los agentes tienen la habilidad de percibir su medio ambiente, que puede ser el mundo físico o un usuario por medio de una interfaz o una colección de otros agentes y responder a los cambios que ocurran en este.

Habilidad social. Los agentes establecen una interacción con otros agentes y posiblemente con humanos, utilizando alguna clase de lenguaje de comunicación entre agentes.

Cooperación. Para llevar a cabo sus tareas, generalmente un agente es considerado como un sistema de cómputo que además de las características anteriores, se le asocian conceptos que usualmente son aplicados a los humanos, como el conocimiento, las creencias, intenciones, obligación y aún más la emocionalidad. La cooperación entre agentes es un mecanismo por el cual los agentes intercambian su conocimiento, sus creencias y sus planes con el fin de trabajar juntos y resolver problemas complejos, los cuales estarían mas allá de sus capacidades individuales.

Pro-actividad. Los agentes no solamente actúan en respuesta a su medio ambiente sino que estos son capaces de exhibir un comportamiento propio, tomando la iniciativa. Éstos pueden razonar acerca de sus intenciones y creencias y tomar las acciones correspondientes.

Racionalidad. Un agente actúa para alcanzar sus metas y no considera las posibles contingencias al tratar de alcanzar sus objetivos.

Varias otras propiedades son asociadas a los agentes como: la *movilidad* que es la habilidad de un agente para desplazarse a través de una red electrónica. La *veracidad* con la que se asume que un agente no comunica información falsa y la *adaptabilidad* que permite que un agente se adecue continuamente a los cambios de su medio ambiente.

3.2.2 Clasificación de los agentes

Con las propiedades consideradas anteriormente para los agentes, los agentes se pueden clasificar de la siguiente forma:

Agentes autónomos son aquellos que habitan en un medio ambiente dinámico, complejo al cual percibe y entonces actúa de manera independiente en este medio ambiente, realizando una serie de tareas para alcanzar una meta.

Agentes de información son agentes que tienen acceso a muchas fuentes de información y tienen la habilidad de manipular dicha información para responder las preguntas propuestas por el usuario o por otros agentes. Algunas veces se les conoce como agentes de Internet.

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

Agentes de interfaz. Esencialmente los agentes de interfaz dan soporte y proporcionan asistencia al usuario. El agente observa y percibe las acciones realizadas por el usuario en la interfaz, aprende de estas acciones y sugiere una manera mas apropiada de realizar la tarea.

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

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

Agentes móviles. Son procesos con la capacidad de transitar por una red que puede ser local o de cobertura amplia, tal como Internet, interactuando con anfitriones remotos para obtener la información requerida por el usuario o bien para ejecutar una tarea que le ayude a alcanzar su objetivo y retornar a su lugar de origen.

Agentes Reactivos. Los agentes reactivos representan una categoría especial los cuales no poseen modelos simbólicos internos, ellos actúan y responden de la forma estimulo-respuesta para presentar estados del medio ambiente que los rodea.

Éstos agentes están basados en los siguientes tres principios básicos [17]:

- La dinámica de interacción entre estos agentes no inteligentes debe conducir a una complejidad emergente.
- Un agente reactivo generalmente es visto como una colección de módulos, los cuales operan de manera autónoma y son responsables de tareas específicas (sensar, percibe, cálculos, etc.). Las comunicaciones entre estos módulos son mínimas y de bajo nivel

- Un agente reactivo tiende a operar sobre las representaciones que están cerca del sensor de datos, en contraste con las representaciones de alto nivel utilizadas por otro tipo de agentes.

Frecuentemente los agentes reactivos son llamados NMAS (Natural Multi-agent System) NMAS son colonias de agentes que muestran un comportamiento emergente y cooperativo para proporcionar un comportamiento integrado del sistema, asumiendo que los agentes individuales son inteligentes.

Agentes híbridos. Como se observó en los agentes mencionados anteriormente, se tienen ciertas deficiencias para cada uno de ellos, lo que hace necesario maximizar las cualidades y minimizar las deficiencias, Pattie Maes [31] propuso adoptar una aproximación híbrida, que tuviera las cualidades de los paradigmas deliberativos y reactivos.

La conformación de un agente híbrido combina dos o más definiciones de comportamiento de agentes en un solo agente. Los agentes híbridos pueden presentar características móviles, de interfaz, colaborativas etc. Una implementación interesante de esta aproximación híbrida es la arquitectura de agente InteRRap desarrollada por (DFKI).

Con el fin de hacer más explícito el alcance y las limitaciones del paradigma orientado a agente. A continuación se hace una descripción detallada de este paradigma, en función de los principios, causas, propiedades (su filosofía) y del soporte que lo definen. La Fig. 3.2 presenta los principales aspectos que caracterizan al paradigma.

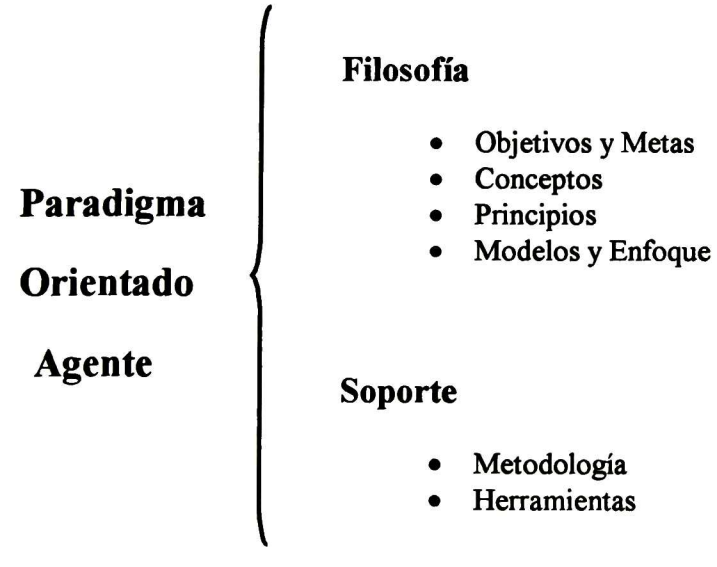


Fig. 3.2 El paradigma de Agente

El paradigma tiene por una parte un enfoque evolutivo ya que toma algunos conceptos ya definidos por la ingeniería de software tradicional. Por otra parte el paradigma tiene un

enfoque revolucionario, ya que muestra una manera distinta de abordar y conceptuar la solución de problemas.

3.2.3 La filosofía del paradigma

La filosofía es considerada como la doctrina que impone como único criterio de valoración de todo principio teórico sus efectos, consecuencias o resultados prácticos, es decir que el contenido inteligible y la evaluación de una experiencia únicamente consisten en los efectos que se obtienen de ella. En este trabajo se considera la filosofía como el Sistema de principios que se establecen para explicar o agrupar ciertos hechos. A continuación de describe para este paradigma, cada uno de los conceptos mostrados en la Fig. 3.1.
Uno de los objetivos principales del paradigma orientado a Agente consiste en abstraer la solución de un problema descomponiendo este problema en múltiples partes en donde cada una de estas partes tiene las propiedades definidas anteriormente para un agente. A este concepto lo llamamos *Abstracción granular*. Una vez que el problema ha sido particionado en componentes, estos componentes se asocian en agrupaciones funcionales. En las agrupaciones funcionales se dan *interacciones entre sus componentes*, las que pueden ser internas o externas, Estas interacciones ocurren por medio de un lenguaje de alto nivel basado en el concepto de speech-act, formando de esta manera una *organización social*.

Objetivos

- Abstracción granular
- Interacción entre sus componentes
- Organización social
- Soluciones Cooperativas
- Validación de las soluciones obtenidas

Como una fase consecutiva, los componentes en interacción se organizan, compartiendo capacidades y recursos con el propósito de llegar a la solución de un problema en *forma cooperativa*. Otro objetivo básico del paradigma consiste el *verificar que las soluciones obtenidas sean las esperadas*, esto se logra por medio de un formalismo basado en la lógica temporal modal, con el cual se especifica el problema a resolver y posteriormente se comprueba que los resultados obtenidos sean congruentes con la especificación del problema.

Metas

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

La instrumentación de aplicaciones distribuidas es una labor complicada con las propiedades y objetivos que tienen los agentes el encargado de elaborar aplicaciones cuenta con una

herramienta que le permite hacer frente en forma apropiada y con una mayor facilidad *la instrumentación de aplicaciones complejas.*

Dada la naturaleza distribuida del concepto de agente y su propiedad para formar asociaciones de agentes y dado que el paradigma cuenta con mecanismos de verificación se hace factible el *desarrollo de aplicaciones confiables* aún en sistemas abiertos. Lo expuesto anteriormente *facilita la instrumentación de aplicaciones de Sistemas MultiAgente*. Y de esta manera enfrentar los problemas que la industria demanda, es decir, *alcanzar una adecuación en las aplicaciones actuales.*

Un agente es un sistema de cómputo encapsulado que está situado en algún medio ambiente, y que es capaz de ejecutar acciones de manera autónoma y flexible en este medio ambiente con el fin de alcanzar los objetivos para los que fue diseñado.

Conceptos
- Autonomía
- Reactivo
- Situado
- Orientado a metas
- Proactivo

El agente es *autónomo*: tiene el control de sobre sus estado interno y sobre su propio comportamiento. Debe ser *reactivo*: capaz de responder en un tiempo razonable a los cambios que ocurren en su medio ambiente, con el fin de responder a sus objetivos de diseño. Está *situado*: o embebido en un medio ambiente particular, en donde recibe entradas relacionadas al estado de su medio ambiente por medio de sensores y actúa en su medio ambiente por medio de sus efectores. Esta *orientado a metas*: tiene objetivos particulares que cumplir, estos objetivos pueden estar representados en forma explícita o implícita dentro del agente. Es *proactivo*: capaz de adoptar de manera oportuna nuevas metas, así como tomar la iniciativa con el fin de satisfacer sus objetivos de diseño.

Para hacer frente a la complejidad que por naturaleza tiene el software industrial, el paradigma orientado Agente adopta los siguientes mecanismos como sus principios básicos: La *Descomposición*, la técnica más básica para enfrentar los problemas extensos consiste en dividirlo en partes más pequeñas, que sean más manejables cada una de las cuales pueda ser tratada de manera independiente. La descomposición es útil al tratar con problemas complejos ya que limita el alcance del diseñador, es decir, el diseñador en cualquier instante solo considera una porción del problema.

Principios
- Descomposición
- Abstracción
- Organización

La *abstracción* es el proceso en el cual se define un modelo simplificado del sistema, en donde se enfatizan algunos detalles y propiedades, mientras que se suprimen otros. Esta técnica funciona porque el diseñador solo tiene un alcance limitado en su visión de interés. La *organización* es el proceso para identificar operar con relaciones mutuas entre los diversos componentes involucrados en la solución de un problema. La habilidad para representar y especificar relaciones organizacionales ayuda al diseñador a resolver los problemas complejos, permitiendo que un grupo de componentes básicos puedan ser agrupados y tratados como una solo unidad de análisis de alto nivel. También ayudan al proporcionar un medio para describir las relaciones de más alto nivel entre varias unidades.

Cuando se adopta un punto de vista del mundo orientado Agente, de inmediato parece aparente que un solo agente es insuficiente para representarlo. La mayoría de los problemas requieren o involucran muchos agentes, ya sea para representar la naturaleza descentralizada del problema, el múltiple "loci" de control, las perspectivas múltiples o los intereses de competencia. Para lograr esto los agentes necesitan interactuar para alcanzar sus objetivos individuales o para manejar las dependencias que se dan por estar situados en un medio ambiente común, éstas interacción van desde la interoperación semántica simple (habilidad de intercambiar comunicaciones comprensibles), hasta las ricas interacciones sociales (la habilidad de cooperar, coordinar y negociar acerca de un curso de acciones).

Modelos
- Interacción
- Estímulo – Reacción
- Colaboración
- Máquinas Virtuales

Los agentes se comportan en concordancia con el modelo *estímulo – reacción* ya que son entidades situadas en un medio ambiente común y tiene la habilidad de responder en un tiempo razonable a los cambios que ocurren en su medio ambiente. Así mismo los agentes adoptan el modelo de *colaboración* para resolver los problemas en forma cooperativa. Los agentes son entidades que solucionan problemas, claramente identificables con interfaces y fronteras bien definidas, para definir éstas interfaces el paradigma adopta el modelo de *máquinas virtuales*.

Enfoque evolutivo. El paradigma El paradigma de agente puede des avistado como una evolución de los paradigmas precedente, ya que toma algunos de los conceptos y principios de estos paradigmas antecesores y los incorpora a éste.

Enfoque
- Evolutivo
- Revolucionario

Enfoque revolucionario. El enfoque revolucionario de este paradigma esta dado por: *i*). los conceptos de granularidad en el cual un problema se divide en un número de componentes

autónomos los cuales pueden ser operados de manera independiente. *ii)* el concepto de organización que se refiere a la asociación de componentes que pueden ser agrupadas en jerarquías de control o en grupos de componentes del mismo tipo. *iii)* el concepto de interoperabilidad con el cual podemos manejar las interacciones entre los agentes en contextos distribuidos abiertos como la Internet y el WEB y *iv)* el concepto de seguridad, este es un concepto esencial en los sistemas de seguridad crítica como la Internet y el WEB, se refiere a los mecanismos que aseguren que las acciones que se ejecutan dentro del sistema son las acciones esperadas y a las políticas y los hechos que brinden la protección requerida para que la información procesada por el sistema esté protegida contra usuarios no autorizados.

3.2.4 El Soporte

El soporte: toma las herramientas de apoyo que proporcionan a un paradigma los medios tangibles para llegar a su realización, éstas herramientas permiten concretar la filosofía asociada a un paradigma, dando como resultado el desarrollo completo e integral del paradigma en discusión. Este paradigma está soportado por herramientas como los principios de la Inteligencia artificial, los fundamentos de los sistemas distribuidos, los conceptos de los sistemas basados en conocimiento, los fundamentos de la lógica temporal modal, técnicas de Ingeniería de Software, etc.

La metodología: La metodología adoptada en ente paradigma se toma del enfoque de la ingeniería de software Orientada Agente [28], [29], ésta abarca: la especificación, la implementación y la verificación del sistema. La metodología se aborda con mayor detalle en apartado 3.4.

Asimismo el paradigma toma la *tecnología:* de la Internet, de los estándares de comercio electrónico [32], [38], [49], de los frameworks como CORBA [39] y de los desarrollos de asociaciones como FIPA [17], etc. Estas tecnologías se toman como cimientos para el desarrollo de aplicaciones.

3.3 El desarrollo Orientado Agentes

El Software Industrial de calidad es complejo por naturaleza, esta caracterizado típicamente por un gran número de componentes que tienen muchas interacciones. Esta complejidad no es accidental, es una propiedad innata de las tareas para las cuales se utiliza el Software [28].

El papel (rol) del desarrollo Orientado Agente consiste en proporcionar modelos, estructuras y técnicas que hacen más fácil el manejo de la complejidad asociada con los problemas del mundo real. La Fig. 3.3 representa una vista canónica de un sistema complejo.

La naturaleza jerárquica del sistema está expresada por medio de enlaces "compuesto de" Los componentes dentro de cada subsistema están conectados por medio de interacciones frecuentes y las interacciones entre componentes estas expresadas por medio de enlaces Interacciones frecuentes.

Hechas estas observaciones el desarrollo Orientado Agente ha divisado un numero de herramientas poderosas para manejar esta complejidad. Los mecanismos principales son: la descomposición del sistema en partes pequeñas, la abstracción para definir un modelo simplificado del sistema y la organización para manejar las interrelaciones entre los diverso componentes del sistema que se está resolviendo.

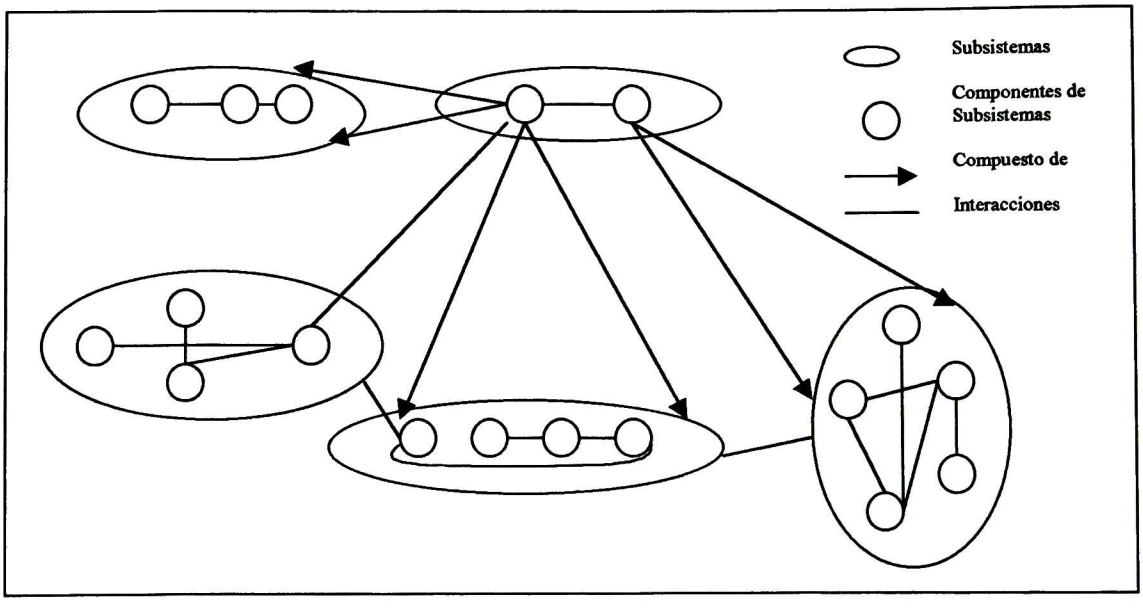


Fig. 3.3 Vista canónica de un sistema complejo

El desarrollo de Software Orientado Agente implica la descomposición del problema a resolver, en múltiples componentes autónomos en interacción, caracterizados por sus roles, responsabilidades y servicios, los cuales tienen objetivos particulares por alcanzar [29].
La abstracción de modelos que definen el comportamiento Orientado Agente, engloba: Agentes, interacciones y organizaciones.

3.4 La Metodología

La metodología para el análisis y diseño de sistemas MultiAgente agrupa un conjunto de conceptos, modelos y procedimientos, para proporcionar el modo de obtener una solución apropiada a los problemas basados en los conceptos y principios englobados en el paradigma de agentes.

La metodología propuesta consta de las siguientes etapas básicas: 1) *Análisis*; en ésta parte se consideran los aspectos y procedimientos para precisar "el que" se quiere resolver; es decir, en primera instancia debemos definir el problema que se ha de resolver. Una vez definido el problema ha resolver, se propone la construcción de un modelo conceptual que involucre los conceptos necesarios para representar una solución intuitiva del problema. Posteriormente se proponen mecanismos para delimitar el ámbito del problema, identificar, clasificar y organizar

a los agentes para elaborar un modelo que represente una organización jerárquica de la solución del problema, para posteriormente utilizar la herramienta de modelado UML para crear un modelo de agentes que represente la solución estática del problema a resolver. 2) *diseño*; en ésta parte de la metodología se agrupan los modelos que representan la solución del problema a resolver; la etapa de diseño de la metodología considera tres modelos ortogonales para representar la solución del problema a resolver: Un modelo estático, representado por el modelo de Agentes; un modelo dinámico representado por el protocolo de interacción y un modelo funcional, representado por la arquitectura del agente.

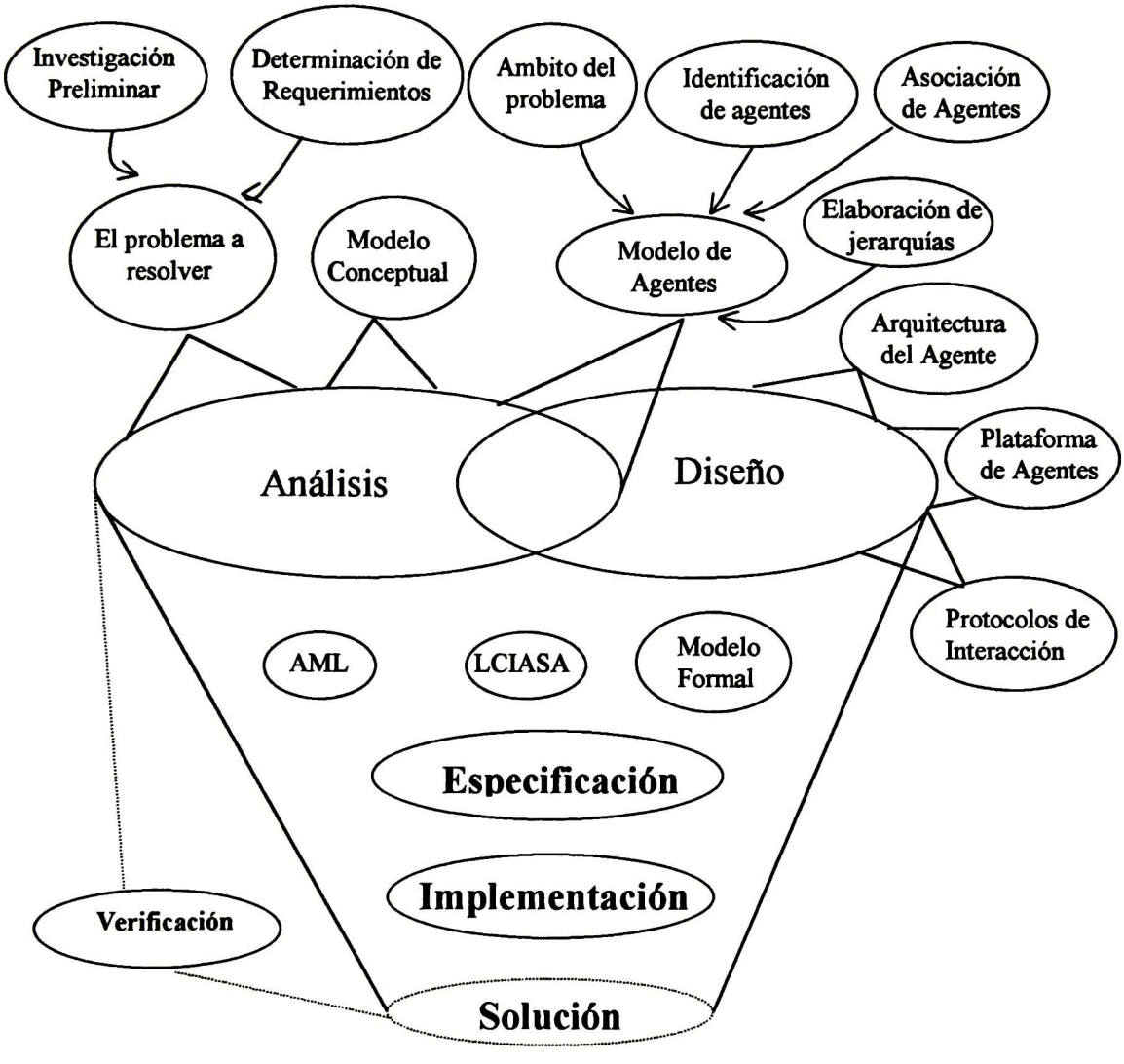


Fig. 3.4 Metodología para el Análisis y Diseño de SMA

3) *Implementación*; una vez que obtenemos el modelo que representa la solución del problema ha resolver se proponen mecanismos para trasladar este modelo en una representación computacional, es decir, obtener una representación ejecutable de la solución del problema. En esta parte se construye la plataforma de agentes como una alternativa para implementar el problema a resolver. 4) *verificación*; el proceso de verificación se aplica durante todo el proceso de desarrollo de la aplicación, con el propósito de asegurar que la solución obtenida sea la esperada, es decir, que se obtenga una solución confiable y coherente en concordancia con el propósito definido en la especificación del problema. El proceso de verificación utiliza los principios y conceptos definidos en el metalenguaje AML, en el lenguaje LCIASA y en los protocolos de interacción para cumplir con su propósito. La fig. 3.4 ilustra los conceptos y modelos básicos asociados a la metodología.

3.4.1 El proceso para el desarrollo de SMA

El proceso del software define la estructura de tareas básicas que se requieren para construir Sistemas de información Orientados Agente con la calidad requerida. A continuación se hace un esbozo del proceso para la Metodología cimentada en el paradigma Orientado Agente.

- ***Especificación:***

  Perspectiva de diseño:

  - Desarrollar una declaración del ámbito del problema.
  - Identificar los roles funcionales y organizacionales del dominio de aplicación.
  - Identificar las responsabilidades, tiempo de vida y servicios asociados a cada rol reconocido anteriormente.
  - Elaborar una jerarquía de subsistemas de agentes.
- Construir el modelo de agentes.
- Desarrollar el modelo formal de agentes.

- ***Implementación:***

  Definir la arquitectura del agente.
  Concretar el modelo de interacción entre agentes.
  Construir la plataforma de Agentes.
  Desarrollar la aplicación.

- ***Verificación:***

  Verificación del ciclo de vida.
  Verificación formal.

La metodología propuesta adopta un grupo de conceptos para definir el modelo que representa el *proceso* de desarrollo de los sistemas MultiAgente. A este modelo también lo denominamos *ciclo de vida* de la metodología, como se muestra en la Fig. 3.5; el modelo evoluciona y progresa con el análisis, diseño, implementación y pruebas de verificación. Este modelo opera en tres diferentes niveles de abstracción:

i) *Primero*, se toma un punto de vista externo, en donde los agentes son modelados como objetos complejos, caracterizados por sus propósitos, sus responsabilidades, por los servicios que llevan a acabo, por la información que requieren y mantienen y por su interacción externa. Los aspectos del punto de vista externo son tratados en los modelos englobados tanto en la fase de especificación como en la fase de implementación, como lo muestra la Fig. 3.4

ii) *Segundo*, se aborda un punto de vista Interno, en donde se muestran las estructuras de datos asociadas a los componentes del agente. Asimismo se establecen la forma en que estos componentes están relacionados para exhibir la funcionalidad del agente. Este punto de vista interno corresponde a los elementos de la arquitectura de un agente en particular, la cual es definida en la fase de implementación.

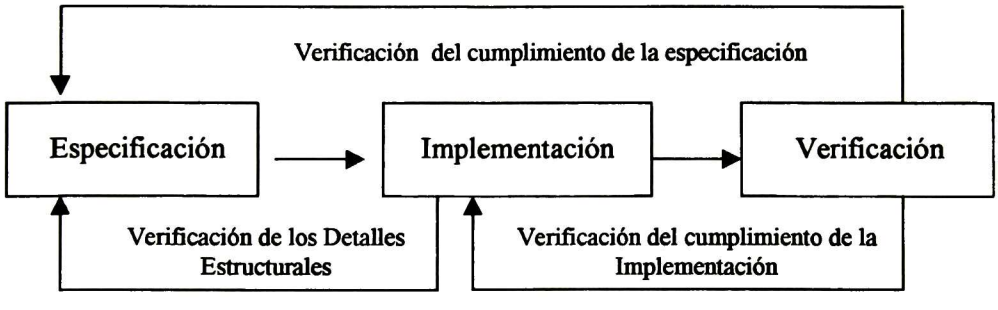


Fig. 3.5 Fases del proceso para la MSMA

iii) *Tercero*, a través del desarrollo de las distintas fases del proceso la metodología propuesta en la Fig. 3.5, es necesario mostrar que el sistema es correcto con respecto a la especificación original. Este nivel de abstracción propone los siguientes mecanismos de verificación: verificación de ciclo de vida y verificación formal.

3.4.2 Estrategia de solución

La metodología consiste de una secuencia ordenada de pasos para el análisis y diseño de problemas basados en el paradigma de agente, con la cual es posible estudiar la solución de estos problemas, desde la definición del problema, pasando por la aplicación de procedimientos de verificación hasta su implementación.

A continuación en la Fig. 3.6 se muestra de manera explícita la secuencia ordenada de los pasos principales incorporados en las fases del proceso de la metodología, mostradas previamente en la Fig. 3.5.

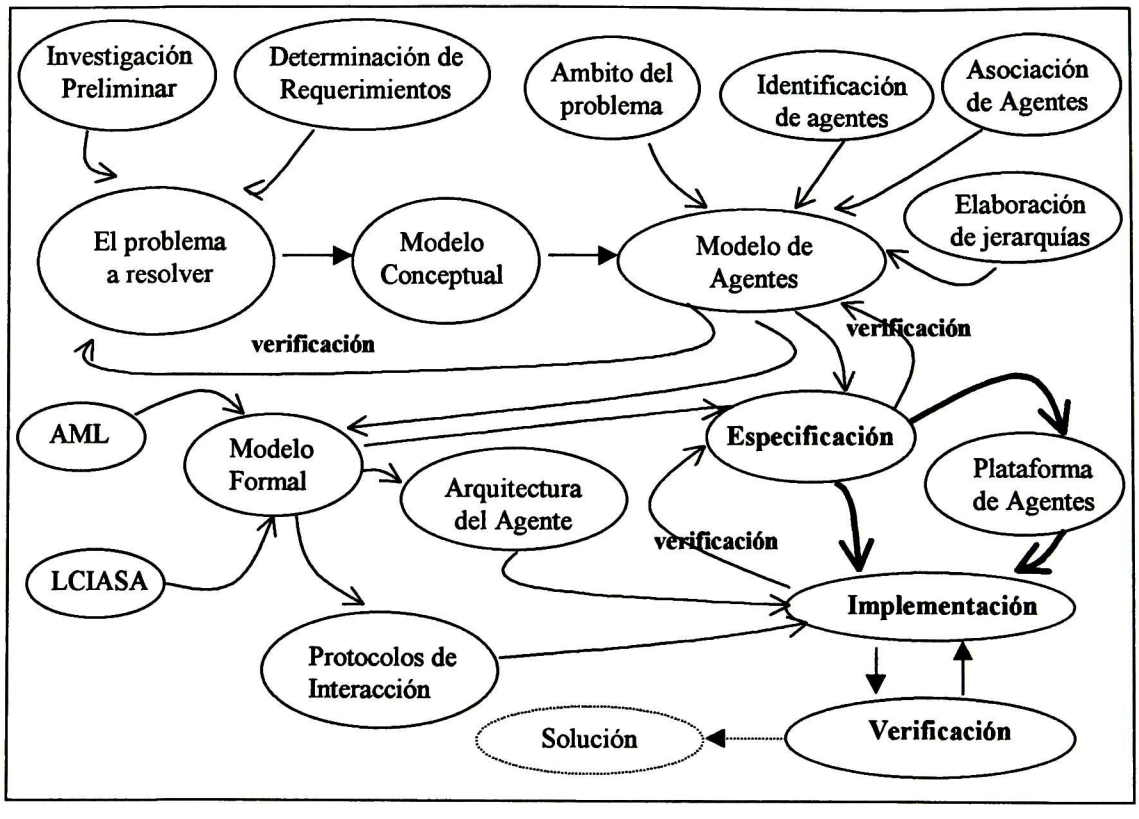


Fig. 3.6 Estrategia de solución

La Fig. 3.7 muestra un esquema para representar los detalles relevantes las fases del ciclo de vida definidas para la MSMA, exhibiendo de manera clara el modo de resolver con orden un problema específico. A continuación se describe brevemente la función de cada uno de los modelos agrupados en las diferentes fases del ciclo de vida de MSMA.

Perspectiva de diseño: Como entrada la perspectiva de diseño recibe primero: la descripción del ámbito del problema que hay que solucionar. En la definición del ámbito del problema se identifican los roles de los agentes, los servicios que ofrecen y proporcionan y sus reponsabilidades. Segundo: Un conjunto de modelos especializados. Estos modelos especializados proporcionan la ayuda necesaria para, obtener modelos precisos, comprensibles y correctos en los cuales se engloban los aspectos relevantes del SMA a resolver.

modelo de agentes: Éste modelo captura la estructura estática del sistema, mostrando los agentes involucrados en el sistema, las relaciones existentes entre ellos y los atributos que caracterizan a cada subsistema de agentes.

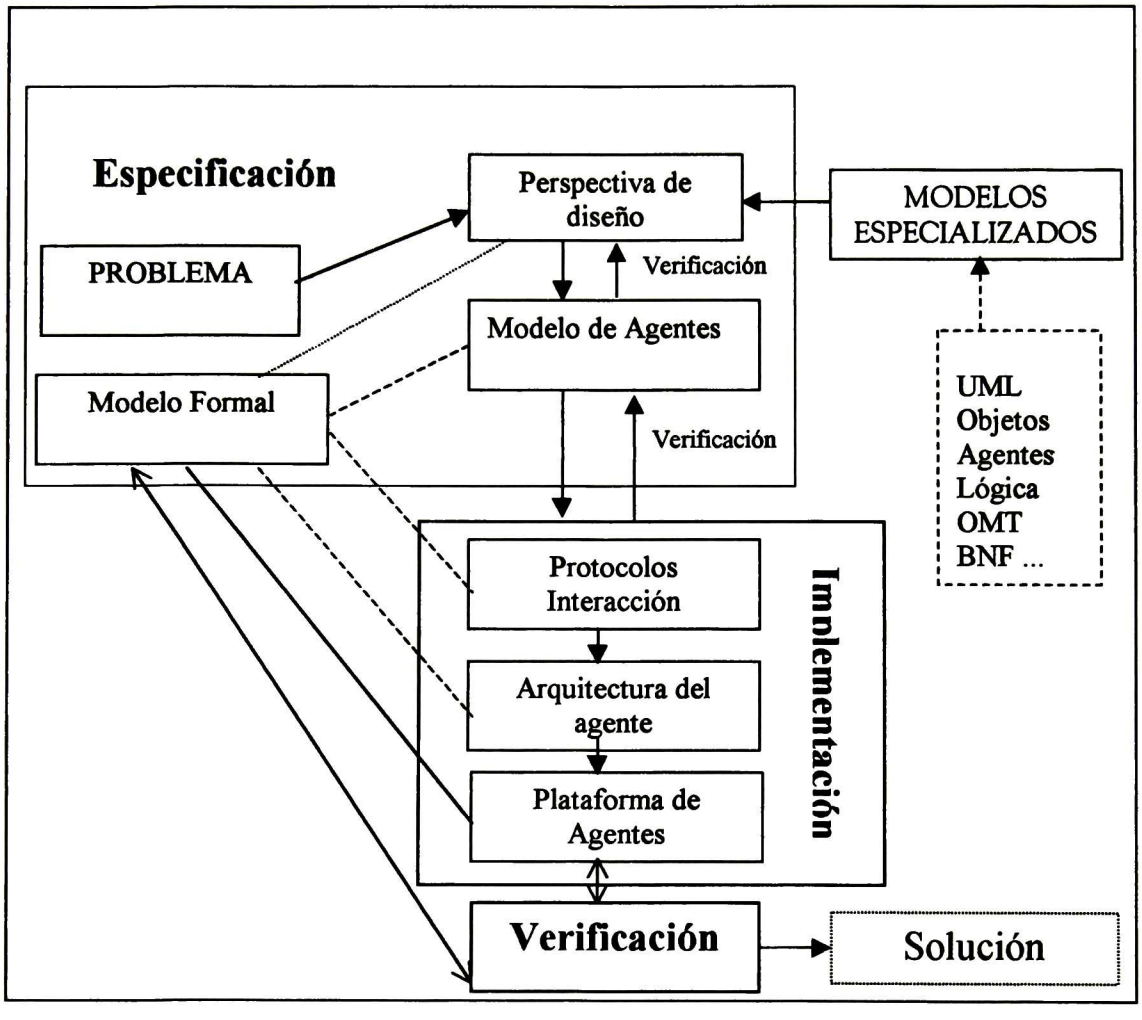


Fig. 3.7 Esquema de la metodología propuesta

Modelo formal: Haciendo referencia al modelo de agentes, se construye un modelo basado en la lógica temporal. Con este modelo se especifica formalmente el SMA. Este modelo formal contiene un Metalenguaje (AML) para dar una interpretación semántica a las acciones de los agentes. También tiene asociado un lenguaje objeto (LCIASA) con el cual se da una interpretación computacional al modelo formal. Es decir LCIASA proporciona la estructura sintáctica y semántica del SMA especificado.

Arquitectura del agente: La arquitectura del agente describe la funcionalidad del SMA. Es decir, aquí se describen los medios para alcanzar las metas establecidas, las estructuras de control, la forma en que ocurren los cálculos dentro del sistema, etc.

Interacción de agentes: La interacción de agentes abarca los aspectos del sistema que están relacionados con el intercambio de información, las políticas de comunicación entre los agentes y los protocolos utilizados para la conversación de los agentes.

En la fase de Especificación se obtienen: los requerimientos del sistema, se definen las propiedades que el sistema es capaz de representar y se define el sistema a resolver. En esta fase se agrupan los modelos: Modelo de agentes, Modelo Formal y Los protocolos de Interacción, Para cumplir con las funciones mencionadas anteriormente. Como se ilustra en la Fig. 3.7. La fase de especificación se aborda con mayor amplitud en el capítulo 4.

En la fase de implementación se estudia el cambio de la especificación abstracta a un sistema computacional concreto. Esta fase agrupa la arquitectura del agente y la plataforma de agentes, como se ilustra en la Fig. 3.7. Esta fase se describe en el capítulo 5.

Finalmente en la fase de verificación se utiliza el modelo formal, en donde se presenta un metalenguaje de agentes basado en la lógica temporal de creencias [55] y se utiliza el lenguaje objeto LCIASA para dar una interpretación al sistema especificado. Esta fase se describe con mayor detalle en el capítulo 6.

Conclusión

La metodología propuesta ofrece un enfoque innovador y original para representar el proceso de solución de problemas basados en el paradigma de agente. Asimismo la metodología propone un proceso que proporciona una solución natural y amigable a los problemas que demanda la industria en la actualidad.

Capítulo 4

Fase de especificación

Resumen

La fase de especificación tiene los siguientes propósitos principales: comprender el sistema que se va a resolver; *determinar las características del sistema*, incluyendo la información que el sistema debe producir y las características operativas como lo son: controles de procesamiento, tiempos de respuesta y métodos de entrada salida; Identificación de las actividades y procesos que permitan *definir el sistema* a resolver. En esta metodología se fusionan los conceptos de análisis y diseño de sistemas de información en una sola fase a la que se denomina especificación, tal como lo hace Jackson en su metodología JSD, como se discutió previamente en la sección 2.2.2.

4.1 Introducción

La fase de especificación abarca los siguientes aspectos: La perspectiva de diseño, el modelo de agentes, el modelo formal. En esta fase se determina el problema a resolver; se definen los requerimientos del sistema; se desarrolla una declaración del ámbito del problema; asimismo

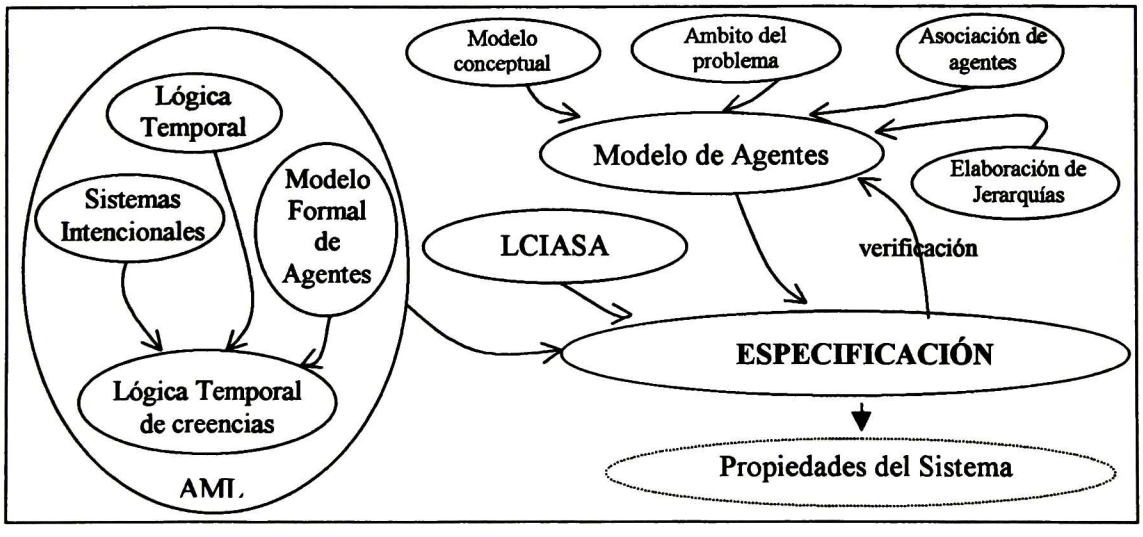


Fig. 4.1 Fase de especificación

se identifica a los agentes involucrados en la solución del problema; también se identifican los roles funcionales y organizacionales del dominio de la aplicación; se crea una estructura jerárquica denominada modelo de agentes, la cual representa los agentes involucrados en la solución del problema así como las relaciones que se dan entre estos agentes. Las actividades

mencionadas anteriormente tienen como propósito definir las propiedades del sistema a resolver. La Fig. 4.1 muestra los conceptos y procedimientos involucrados en esta fase de la metodología.

Lo expresado anteriormente tiene como fin de establecer en primera instancia el "que" se va hacer y después se define el modelo de agentes, el modelo formal y los protocolos de interacción para determinar el "como" se va a resolver el sistema propuesto.

4.2 Perspectiva de Diseño

En la perspectiva de diseño debemos conocer que es lo que va a resolver y como se va a resolver. Para esto se definen las características que el sistema debe exhibir y se identifican los Agentes involucrados en el problema a resolver; la forma en que los agentes están organizados y las actividades que se llevan a cabo dentro del sistema, con el fin de poder definir explícitamente el problema que se va a resolver.

Como punto de partida la perspectiva de diseño recibe la definición del problema a resolver, mediante un proceso de *abstracción*: se identifican el ámbito y las restricciones del problema; después, por medio de la *descomposición* de las funciones de cada agente, se definen los mecanismos para reconocer y asociar a los agentes involucrados en la solución del problema; luego se elaboran las jerarquías de subsistemas de agentes y finalmente, mediante un proceso de *síntesis*, se crea el modelo de agentes que representa la estructura estática, en donde de agrupan, los diversos componentes involucrados en la solución del problema a resolver, y en donde también se define la forma que éstos componentes son relacionados entre ellos.

Para ilustrar el proceso de la metodología propuesta, se propone resolver un problema de comercio electrónico, en particular el problema consiste en desarrollar un sistema de cómputo que haga viable la compra y venta de artículos utilizando las facilidades y servicios ofrecidos por la Internet y el web. Este sistema le proporciona al usuario las facilidades para especificar las condiciones del artículo que desea comprar, así como las opciones de compra más convenientes, las cuales son ofrecidas por los posibles vendedores.

Este mismo problema será utilizado a lo largo de este trabajo de tesis con el fin de ejemplificar por medio de diferentes modelos y esquemas, los aspectos de cada una de las fases descritas en la metodología que se propone.

4.2.1 Declaración del ámbito del problema

En la declaración del ámbito del problema, se hace un reconocimiento de los elementos básicos del problema, tal como lo percibe el usuario, el diseñador y el cliente. La idea fundamental de esta fase consiste en definir con claridad el problema a resolver así como sus alcances y limitaciones, con el fin de eliminar ambigüedades y discrepancias en los objetivos y metas que se persiguen para alcanzar para el problema propuesto.

La fig. 4.2 ilustra de manera gráfica un modelo conceptual, en donde se muestran algunos de los elementos componentes del problema a resolver, en esta figura se muestra un modelo de comercio electrónico en donde un comprador desea adquirir un producto, para ello cuenta con una página del comprador que contiene una lista de proveedores.

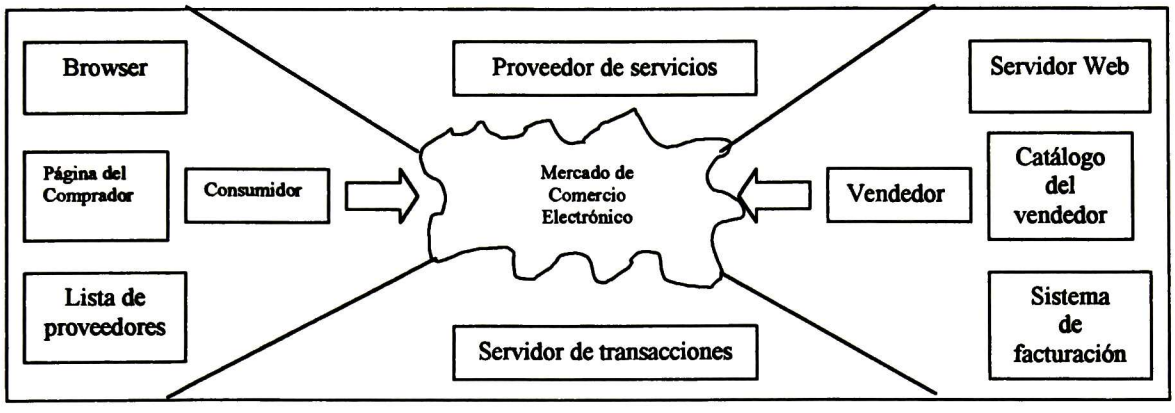


Fig. 4.2 Modelo conceptual para la aplicación de Comercio Electrónico

Para lograr una definición clara de ámbito del problema se siguen los pasos especificados en los métodos tradicionales de análisis y diseño de sistemas de información [36], [46], tales como: *la investigación preliminar, determinación de requerimientos del sistema*, etc. El modelo representado en la fig. 4.1 se derivó a partir del proceso de abstracción del problema a resolver, en donde se define un modelo simplificado del sistema a resolver, el cual enfatiza algunos detalles o propiedades, mientras que suprime otros.

El proceso básico de compra - venta de artículos en el modelo de la fig. 4.1 es el siguiente: El comprador utiliza un Browser para contactar a los compradores, revisa el catálogo de los vendedores, elige un producto de un posible vendedor, hace un pedido, el vendedor le envía una petición OBI al comprador, el comprador aprueba la orden y el vendedor le surte el pedido.

Una vez definido el problema a resolver y que se han identificado los componentes básicos involucrados en el modelo que propone una solución del problema, a continuación se hace de una manera breve la descripción de los alcances y limitaciones del problema a resolver:

Alcances del comprador.

- El sistema debe facilitar al comprador configurar las opciones de compra para adquirir el articulo deseado.
- El comprador debe contar con un browser que le proporcione acceso al web (por medio de sus direcciones URL), asimismo recibir, interpretar y mostrar páginas web.

Alcances del vendedor

- El vendedor debe contar con un catálogo del vendedor en donde se muestra la información de los productos que este ofrece.
- El vendedor esta habilitado para ofrecer entre otros, los servicios de ofertas y propaganda al comprador
- El vendedor debe contar con un servidor de páginas web, que reciba y envíe la información solicitada por el browser.
- El vendedor cuenta con un proveedor de servicios que ofrece los servicios creación y acceso de sitios web, actualización de catálogos y servicios de comercio (Autorización de crédito, etc); Servicios de facturación; provisión de ambientes seguros para realizar las transacciones.

Limitaciones del problema:

- El problema a resolver solo se limita a las operaciones de compra y venta de artículos al menudeo, es decir las operaciones comerciales se dan entre un comprador final y una empresa (modelo B2C)

4.2.2 Identificación de los agentes

La identificación de los agentes se lleva a cabo tomando como apoyo el contexto del problema, reconociendo las entidades que tengan ciertas responsabilidades, roles funcionales y de organización dentro del dominio de aplicación del problema.

Para facilitar la identificación de los agentes se utiliza una tabla en la que se visualicen los aspectos de mayor relevancia, con los cuales los agentes puedan ser caracterizados. A esta tabla le llamaremos: *página genérica de agentes* (Pg), o simplemente página de agentes. La Tabla 4.1 exhibe un esquema genérico para Pg. Como se observa en la fig. 4.1 la página de agentes Pg está definida por un conjunto de atributos, características o propiedades, con los cuales se hace una caracterización estructural y funcional de los agentes.

Agente	Responsabilidades	Rol	Servicios que proporciona	Es-parte-de	Organización externa	Propósito	...	Dirección URL	Relacionado con
Id-A1							...		
Id-A2							...		
Id-A3							...		
...	...	...	...	...	...	...	...		...
Id-Aj							...		

Tabla 4.1 Esquema genérico de la Página de Agentes Pg.

Para facilitar la identificación de los agentes se utiliza una tabla en la que se visualicen los aspectos de mayor relevancia, con los cuales los agentes puedan ser caracterizados. A esta tabla le llamaremos: *página genérica de agentes* (Pg). La Tabla 4.1 exhibe un esquema genérico para Pg.

Como se observa en la tabla. 4.1 la página de agentes Pg está definida por un conjunto de atributos, características o propiedades, con los cuales se hace una caracterización estructural y funcional de los agentes.

Para obtener una representación más expresiva y conveniente con la cual representar y manipular las páginas de agentes y la información que estas contienen, se utilizan las siguientes convenciones:

- Las páginas y páginas de agentes se denotan por
  Nombre-de-Página = < $Atributo_1$, $Atributo_2$, $Atributo_3$, ... $Atributo_j$>
- Los nombres de las páginas y páginas son únicos.
- Para lograr el acceso a la información de una página o sub-página de agentes se utilizan las siguientes operaciones del álgebra relacional: la proyección $\Pi s^{(Pag)}$, la selección $\sigma_p{}^{(Pag)}$, el producto cartesiano (Pag1 X pag2), la unión (Pag1 $\cup$ Pag2), la diferencia (Pag1 – Pag2) y las operaciones adicionales como el producto natural Pag1 |X| pag2, la asignación, etc. Las cuales son derivadas de estas operaciones fundamentales. En donde P es un predicado y S es una lista que consta de algunos de los atributos de la página de agentes Pag.

Utilizando las operaciones mencionadas anteriormente, la tabla 4.1 puede dividirse por conveniencia en subtablas o sub-páginas, (a las cuales, también se les llamará simplemente páginas) con el fin de particularizar algunas de las propiedades de los agentes, cumplir con una función específica. Para hacer más eficiente y apropiado el almacenamiento y manejo de la información que contienen las páginas, se toman en cuenta los aspectos de *normalización*, empleados en el contexto del diseño de las bases de datos relacionales [30].

El objetivo principal de las páginas consiste en hacer una descripción explícita de las propiedades funcionales y organizacionales de los agentes involucrados en el sistema. Las páginas de agentes deben contener la información suficiente para proporcionar una descripción de los aspectos relevantes relacionados con al sistema a resolver. Las páginas de agentes almacenan detalles y descripciones de los agentes, los roles del agente, las responsabilidades del agente, los servicios ofrecidos y proporcionados por los agentes, sus direcciones URL, etc. Con los cuales es posible caracterizar, especificar, implementar y validar el sistema de agentes derivado del problema a resolver.

La página de agentes contiene información a la que se hace referencia a lo largo de todas las fases de la metodología, por esta razón se considera a la página de agentes como uno de los conceptos centrales para la metodología propuesta.

4.2.2.1 Identificación de propiedades de los agentes

Las responsabilidades, tiempo de vida, roles, acciones, organización y servicios asociados a cada uno de los agentes son derivadas de los requerimientos del dominio del problema. La descripción completa de las propiedades de los agentes se hace por medio de la tabla

mostrada en la tabla 4.1. Como puede observarse, ésta tabla con frecuencia es muy extensa, por esta razón en la práctica se manejan páginas que describen a conveniencia o necesidad del usuario o analista, algunos aspectos y propiedades de ésta tabla genérica.

A continuación, a manera de ejemplificar la creación de páginas de agentes, en la fig. 4.2 se presentan una página en donde se describen algunas propiedades de los agentes identificados en el contexto del problema a resolver, señalando las responsabilidades, los roles y las relaciones con otros agentes.

Agente	Responsabilidades	Roles	Relacionado con
Página del comprador	Proporciona una lista de proveedores Captura opciones de compra Localización de productos	Utiliza browser Proporcionar listados Emplea OBI	Browser Lista de proveedores Servidor OBI del Comprador
Browser	Proporciona acceso a web Contacta con servidores Web Recibe, interpreta y muestra páginas HTML	Contacta servidores Desplegar páginas	Página del comprador Servidor OBI del comprador
Lista de proveedores	Almacenar la lista de proveedores Actualizar lista de proveedores Registrar operaciones ejecutadas	Actualiza lista Anuncia proveedores	Página del comprador Proveedor de servicios Servidor OBI del Comprador
Servidor OBI del comprador	Recibir peticiones del vendedor Enviar peticiones al vendedor	Envía petición Recibe petición	Browser Servidor de transacciones
Proveedor de servicios	Creación de sitios web Actualización de catálogos Seguridad Sistemas de enlace	Proporciona servicio	Lista de proveedores Catálogo del vendedor Servidor de transacciones
Catálogo del vendedor	Acceso a la página Lista de artículos Desplegar condiciones de venta	Presenta producto	Lista de proveedores Proveedor de servicios Servidor de transacciones
Servidor de transacciones	Sistema de seguridad Sistema de facturación	Envía transacción Recibe transacción	Proveedor de servicios Servidor OBI del Vendedor Servidor OBI del comprador
Servidor OBI del vendedor	Recibir peticiones del vendedor Enviar peticiones al vendedor	Envía petición Recibe petición	Browser Servidor de transacciones
Sistema de facturación	Cálculo de impuestos Autorización de crédito	Impuestos Autoriza	Proveedor de servicios Servidor de transacciones
Sistema de seguridad	Autenticación Cifrado de información	Autentifica Cifra	Proveedor de servicios Servidor de transacciones

Tabla 4.2 Página de responsabilidades, roles y relacionado con

Utilizando la notación dada anteriormente, la tabla 4.2 puede expresarse por medio de la notación del álgebra relacional, de la siguiente manera.

$SP_{\pi\pi} = \Pi s^{(Pg)}$ en donde s = <responsabilidades, roles, relacionado-con > y pg es la página genérica de agentes.

4.2.3 Asociación de los agentes

La asociación de los agentes a la que también se le llamará *organización de agentes* se hará tomando como base el método de categorización clásica, el cual pregona que todas las entidades que tiene una determinada propiedad o colección de propiedades en común forman una categoría [54].

Considerando este método de asociación, los agentes se organizan tomando en consideración sus propiedades: responsabilidades; roles, manifestados por el propósito o capacidad por la que un agente se relaciona con otro agente; acciones que realiza; servicios que suministra; etc.

La asociación de agentes puede llevarse a cabo haciendo hincapié en las responsabilidades que denotan el conocimiento que el agente tiene y por las acciones que el agente puede realizar. Las responsabilidades están encaminadas a comunicar una expresión del propósito de un agente y su lugar dentro del sistema. Las responsabilidades de un agente son todos los servicios que éste suministra para todos los contratos que soporta.

La asociación de agentes en subsistemas se lleva a cabo utilizando la propiedad "es-parte de" en la cual los agentes que representan componentes para alcanzar algún propósito o función definida se asocian a un agente que tiene éste propósito y el cual representa el ensamblaje completo de estos agentes.

Para ejemplificar las asociaciones se presenta la tabla que corresponde al subsistema de seguridad en donde se identifican cuatro agentes con sus responsabilidades correspondientes como se muestra en la tabla 4.3

Agente	Responsabilidades	Propósito	Es-parte-de
Cifrador	Codifica la información	Seguridad Del Sistema	Subsistema de seguridad
Decodificador	Revela la información		
Autenticador	Autentica usuarios y acciones		
Autoriza-petición	Manejo de las peticiones de Autorización		

Tabla 4.3 Página del subsistema de seguridad

La tabla 4.3 es derivada de la siguiente expresión: $\sigma_p(\Pi_S{}^{(Pg1x1SPrr)})$ en donde s=<agente, responsabilidades, propósito, Es-parte-de > y p = (Agente = Subsistema de seguridad).
Tomando como referencia la información presentada en la tabla 4.3, ahora esta información es representada en forma gráfica, en donde se utiliza la notación definida previamente para esquematizar esta información.

En la fig.4.3 se representa a un grupo de agentes que tienen un propósito común, en este caso, el de realizar tareas encaminadas a mantener la seguridad del sistema. En la figura se observa

como éstos agentes se asocian al agente: subsistema de seguridad, el cual tiene este propósito de proporcionar los componentes necesarios para mantener la seguridad en el sistema.

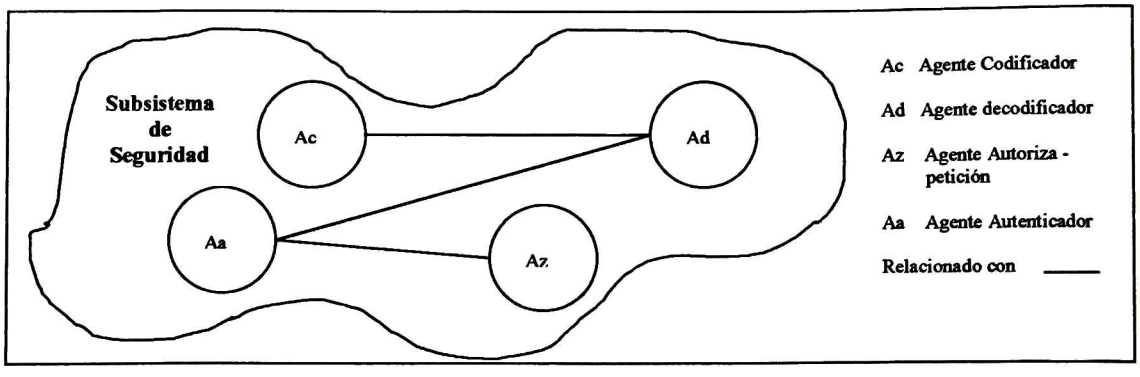


Fig. 4.3 Subsistema de seguridad

En la fig. 4.3 se observa que el agente subsistema de seguridad representa el ensamblaje completo de los cuatro agentes que lo componen, este ensamblaje puede ser visto como una unidad independiente que mantiene cierta autonomía para cumplir con una función específica.

4.2.4 Elaboración de jerarquías de subsistemas de agentes

Una vez identificados los agentes y sus propiedades se hace un análisis de la página de agentes y en función de las propiedades encontradas para cada uno de los agentes, los agentes se agrupan en unidades independientes a las que llamaremos subsistemas, como se mostró en la sección anterior. Después se agrupan las entidades que tienen responsabilidades comunes y se forman jerarquías de subsistemas que involucran a subsistemas de mayor jerarquía que incorporan responsabilidades generales y subsistemas que especializan su comportamiento.

Ahora se busca una relación de más alto nivel, tomando como base asociaciones del tipo *"compuesto de"* como se mostró en la fig. 4.2. Estas asociaciones reflejan cierta funcionalidad particular entre los componentes básicos asociados como subsistemas y agentes de otros subsistemas.

A ésta asociación la llamaremos generalización, la cual es una relación de un subsistema a una o más versiones refinadas otro subsistema; al subsistema que se está refinado se le llama supersistema y a la versión refinada se le llama subsistema. Esta es la manera de identificar las jerarquías en un Sistema MultiAgente, creando estructuras de organizaciones o grupos de agentes que correspondan a jerarquías de control dadas entre agentes u organizaciones de agentes. Para ejemplificar la construcción de las jerarquías, se toma como referencia la página de agentes mostrada en la tabla 4.4

Agente	Responsabilidades	Propósito	Es-parte-de
Servidor de transacciones	Manejo de las peticiones de Autorización. Procesa los certificados digitales. Mantiene un registro de las transacciones. Procesa transacción para múltiples vendedores. Maneja transacciones de crédito y débito. Manejo del proceso de pago	Ofrecer la tecnología completa para realizar transacciones electrónicas y para mantener la seguridad dentro del sistema	Proveedor De servicios
Servidor OBI del vendedor	Recibir peticiones del vendedor Enviar peticiones al vendedor	Enviar peticiones Recibir peticiones	Servidor de transacciones
Sistema de facturación	Cálculo de impuestos Autorización de crédito	Cálculo de impuestos Operar proceso facturación	Servidor de transacciones
Sistema de seguridad	Autenticación Cifrado de información	Autentifica Cifra	Servidor de transacciones

Tabla 4.4 Subpágina de responsabilidades, propósito, Es-parte-de

La tabla 4.4 es derivada de la siguiente expresión: $\sigma_p(\Pi_s{}^{(Pg1x1SPrm)})$, en donde s = <agente, responsabilidades, propósito, Es-parte-de > y p = (Agente = Servidor de transacciones $\wedge$ Es-parte-de = Servidor de transacciones).

Tomando como referencia la información presentada en las tablas 4.3 y 4.4, ahora esta información es representada en forma gráfica, en donde se utiliza la notación definida previamente para esquematizar esta información.

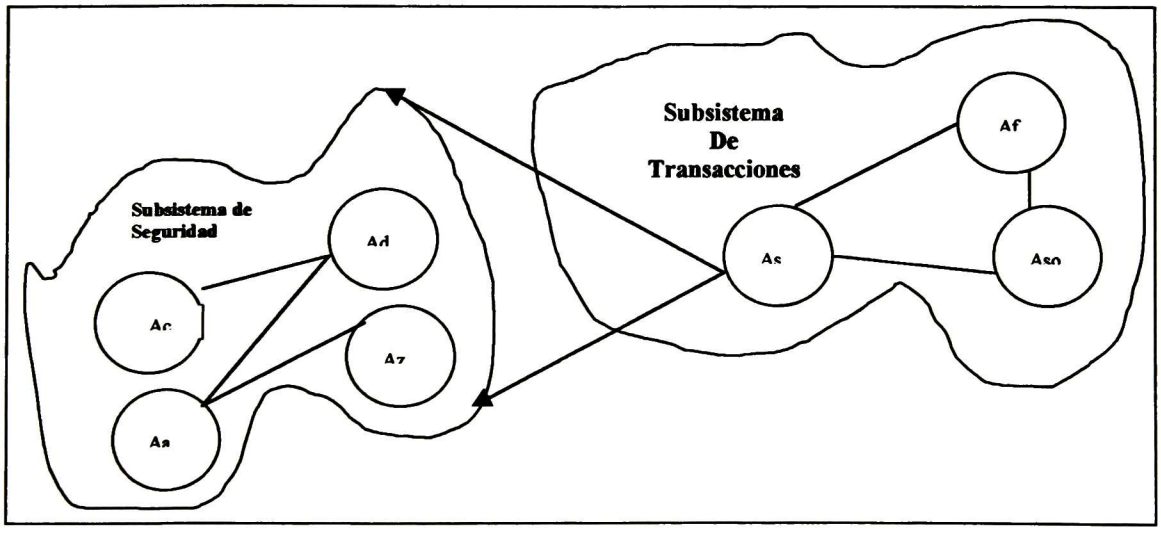


Fig. 4.4 Identificación de jerarquías entre subsistemas.

En la fig. 4.4 se observa que el subsistema de transacciones tiene asociados a los agentes: agente de seguridad As, agente de facturación Af y al agente Servidor OBI Aso, los cuales son

agrupados para cumplir con una función específica y tratados como una unidad de más alto nivel de análisis.

Asimismo, en esta figura se observa que el agente de seguridad As representa por medio de la relación "compuesto de", una relación jerárquica de mas alto nivel con respecto al subsistema de seguridad.

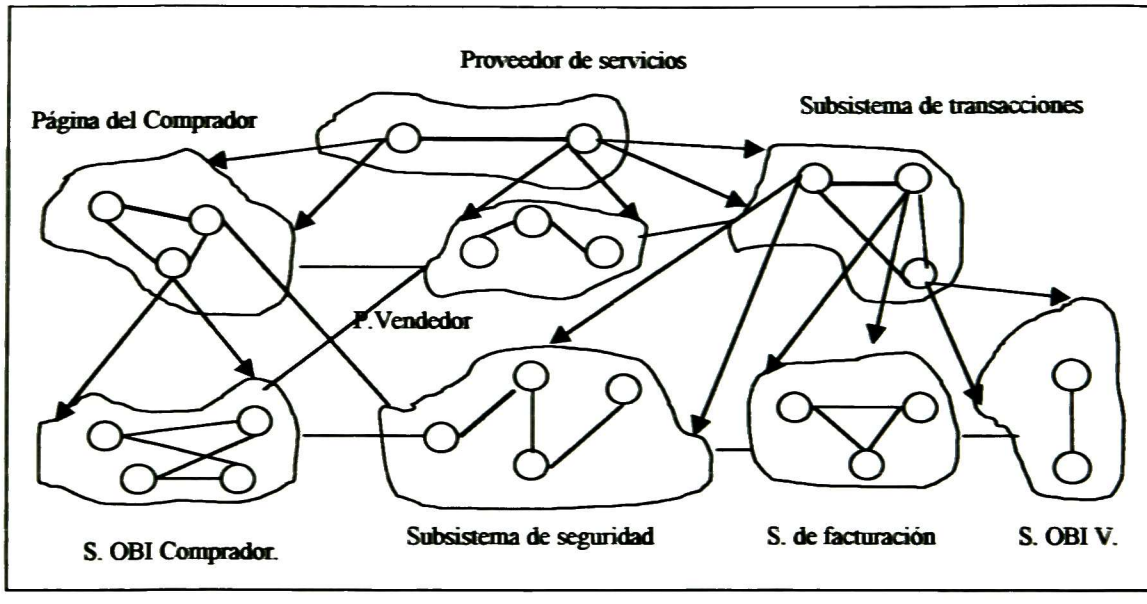


Fig. 4.5 Esquema simplificado de la Organización para el modelo de comercio electrónico

Podemos decir que el subsistema de transacciones es una generalización del subsistema de seguridad. También es posible considerar al subsistema de seguridad como una specialización del subsistema de transacciones.

La fig. 4.5 corresponde al proceso de síntesis, en donde se identifican los componentes y las relaciones existentes entre éstos componentes, agrupándolos en un esquema que representa la estructura global, así como los aspectos relativos a la funcionalidad del problema de comercio electrónico propuesto con anterioridad en la fig. 4.2.

4.3 Modelo de Agente

Una vez que hemos obtenido el esquema que representa la organización del problema a resolver se utiliza una herramienta para modelar esta estructura, de tal manera que el modelo que se obtiene sea más sencillo de entender y que éste modelo obtenido esté apegado a herramientas que proporciones una representación estándar para el modelado de sistemas.

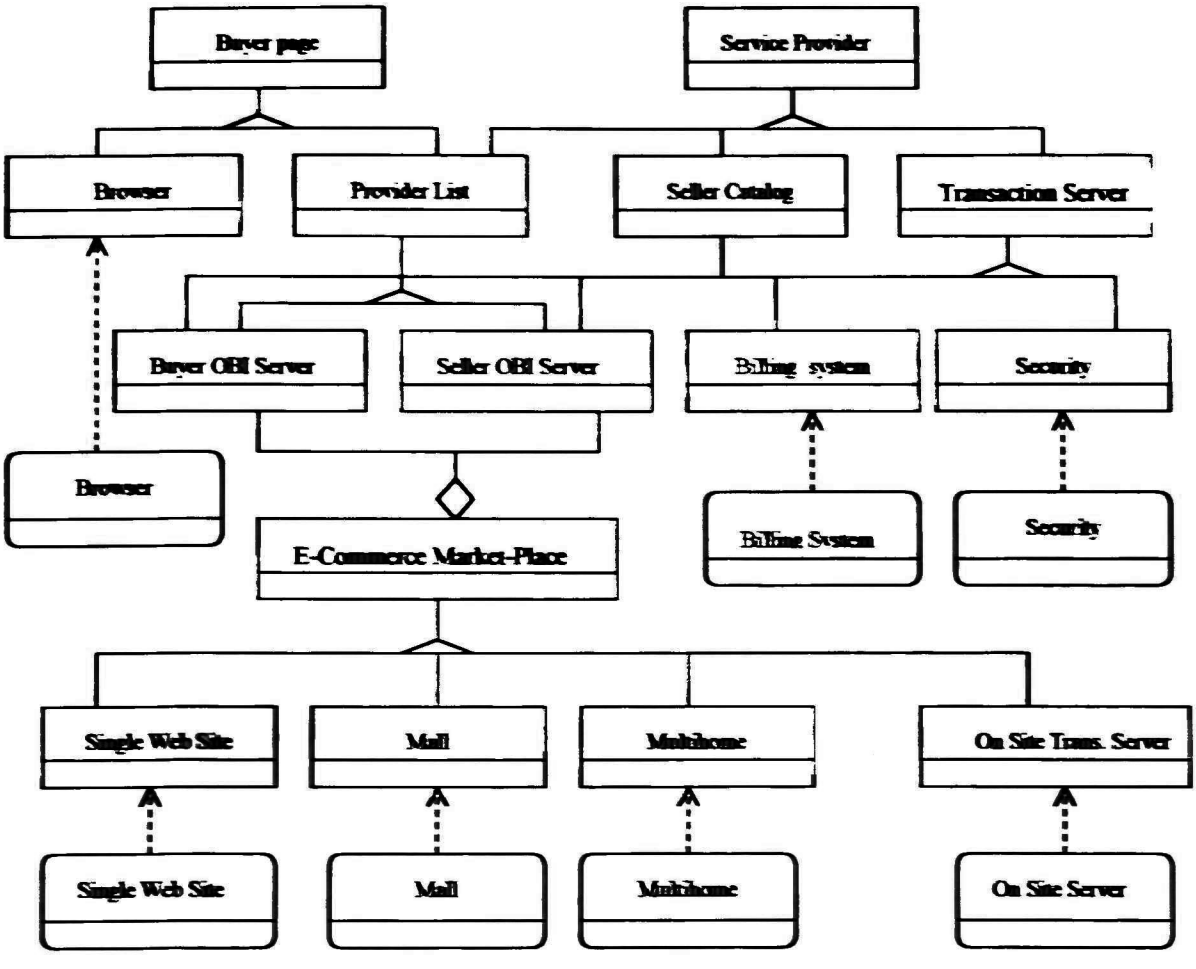


Fig. 4.6 Modelo de agente para la aplicación de comercio electrónico

En este trabajo se utiliza el lenguaje unificado de modelado UML [44] como herramienta para modelar el esquema de la organización de agentes obtenido previamente en la fig. 4.5. Con UML creamos un modelo al cual se designa como *Modelo de agente*. Este modelo captura la estructura estática del sistema, mostrando que agentes están involucrados, las relaciones existentes entre ellos y los atributos que caracterizan a cada agente. El modelo de agente puede verse como un modelo conceptual que es utilizado para hacer una descripción jerárquica de lar relaciones que se dan entre los agentes involucrados en la descripción del problema a resolver. La fig. 4.6 exhibe el modelo de agente para la aplicación de comercio electrónico que se ha planteado anteriormente. El modelo de la Fig. 4.6 contiene clases abstractas y clases concretas o instanciables, éste modelo captura las relaciones de herencia, y asociación que se presentan entre los agentes.

4.4 Modelo Formal

Una vez obtenido el modelo de agentes, es de gran utilidad contar con modelo formal, con el cual sea posible hacer una descripción de las propiedades funcionales y operacionales, así como hacer un estudio de la consistencia que debe exhibir el sistema MultiAgente que se propone resolver, antes de pasarlo a la fase de implantación.

El modelo formal que se presenta en este trabajo de tesis, está constituido por dos partes básicas. Primero: El meta lenguaje de agentes (AML), el cual está basado en la lógica temporal de creencias. Segundo: el lenguaje objeto (LCIASA), el cual está asociado con AML, con el fin de contar con un lenguaje con mayor facilidad de interpretación y así enfrentar la dificultad que representa el uso de la lógica temporal para describir a los agentes y a las acciones que estos realizan.

En esta sección se presenta en primer término el Metalenguaje de agentes y después de presenta el lenguaje objeto LCIASA.

4.4.1 Meta Lenguaje de Agentes

Como se mencionó previamente AML está soportado por los principios teóricos de la lógica, el tipo de lógica utilizado en AML es una clase de lógica no clásica, en donde el modelo del tiempo ofrece las bases para describir en forma dinámica el SMA. A continuación se obtiene un modelo para un SMA y después de presentan los principios de la lógica temporal utilizada en AML.

4.4.1.2 Modelo de un Sistema MultiAgente:

El modelo de un MAS tiene las siguientes propiedades básicas: [55]

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

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

- Los agentes pueden ejecutar acciones: Se infiere que los agentes pueden ejecutar solamente acciones privadas, es decir pueden ejecutar acciones, las cuales operan en su propio estado. Sea **Ac** el conjunto de todas las acciones.
- Los agentes pueden enviar mensajes: Aunque las comunicaciones no se aplican de manera universal en DAI, esta es sin embargo una adjudicación común, entonces se supone que los agentes pueden comunicarse enviando mensajes. Un mensaje es una tripleta $< i, j, \phi >$

en donde $i \in$ **Ag**, representa al transmisor del mensaje, $j \in$ **Ag** es el receptor del mensaje y $\phi \in$ **Form(L)** es el contenido del mensaje. Sea **Mess** el conjunto de todos los mensajes:

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

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

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

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

Con estos conceptos, ahora es posible definir un agente como el cuádruplo:

$$A \overset{\text{def}}{=} < \beta^{\circ}, A_{\text{ç}}, M, \eta >$$

En donde:

$\beta^{\circ} \in BS$ es el conjunto de creencias iniciales del agente
$A_{\text{ç}} : BS \rightarrow Ac$ es la función de Acción del agente
$M : BS \rightarrow \wp(\text{Mess})$ es la función de generación de mensajes del agente
$\eta : BS \times Ac \times \wp(\text{Mess}) \rightarrow BS$ es la función del siguiente estado del Agente.

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

$$\text{MAS} \overset{\text{def}}{=} \{ < \beta^{\circ}_i, A_{\text{ç}\,i}, M_i, \eta_i > : i \in \textbf{Ag} \}$$

El concepto de MAS es útil, ya que en la solución de un problema, es evidente que un solo agente es insuficiente o poco practico para resolverlo. Es posible afirmar sin pérdida de generalidad que la solución de cualquier problema por medio del paradigma Orientado-Agente involucra múltiples agentes

4.4.1.3 El modelo de ejecución

El modelo de ejecución depende de la noción de estado de un sistema y de los cambios en el estado causados por las transiciones. Un estado es una instancia del conjunto de creencias de cada agente en el sistema en algún momento en el tiempo. Un cambio de estado o transición ocurre cuando un mensaje es recibido y se ejecuta una acción por parte de uno o más agentes.
El estado inicial del sistema está dado por el conjunto inicial de creencias del conjunto de agentes, esto es β^0_i
Formalmente, el estado β^1_i del agente i en el tiempo 1 está dado por la ecuación:

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

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

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

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

$$\sigma : S_0 \xrightarrow{\tau_0} S_1 \xrightarrow{\tau_1} S_2 \rightarrow \quad \rightarrow S_u \xrightarrow{\tau_u} \ldots$$

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

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

La lógica temporal es una rama de la lógica modal que trata con el desarrollo de situaciones en el tiempo, es decir describe situaciones dinámicas en donde la interpretación de las fórmulas se realiza usando marcos en los cuales las relaciones de accesibilidad son relaciones lineales [19].
La lógica **LA** es una lógica proposicional y contiene tres operadores atómicos: **Bel** para describir las creencias de un agente, **Send** para describir los mensajes que el agente envía y **Do** para describir las acciones que el agente ejecuta [55].

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

4.4.1.5 Reglas sintácticas para LA

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

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

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

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

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

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

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

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

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

La primera regla trata con fórmulas atómicas o átomos de AL, la fórmula **true** es una constante lógica para denotar verdad, esta siempre es satisfecha. La fórmula **(Bel i, ϕ)** se lee: el agente i cree ϕ. La fórmula **(Do i, α)** se interpreta como: el agente i ejecuta la acción α. La fórmula **Send(i, j, ϕ)** describe el envío de mensajes y será satisfecha si el agente i ha enviado al agente j un mensaje con el contenido ϕ.
Los conectivos proposicionales $\wedge$(and), $\vee$(or), $\Rightarrow$(if...then...), $\Leftrightarrow$(iff) y $\neg$(not) conservan su notación y semántica estándar. La fórmula $O\phi$ es satisfecha si ϕ es satisfecha en el siguiente tiempo (u+1). La fórmula $\otimes\phi$ es satisfecha si ϕ fue satisfecha en el tiempo anterior (u-1).
La tabla 4.5 presenta el significado de los operadores temporales con los cuales es posible escribir e interpretar correctamente expresiones para LA.

Regla	Significado
(Bel *i* ϕ)	El agente *i* cree ϕ
(Send *i j* ϕ)	El agente *i* envió al agente *j* el mensaje ϕ
(Do i α)	El agente i ejecuta la acción α
Oϕ	El siguiente ϕ
$\otimes\phi$	Previamente ϕ
$\Box\phi$	De ahora en adelante ϕ
$\Diamond\phi$	Eventualmente ϕ
$\phi \mu \psi$	ϕ hasta que ψ (no estricto)
$\phi \omega \psi$	ϕ a menos que ψ
$\upsilon\phi$	ϕ hasta ahora
$\blacklozenge\phi$	ϕ alguna vez
$\alpha\phi$	ϕ antes
$\phi\beta\psi$	ϕ Cuando ψ
$\phi \xi\psi$	ϕ desde que ψ

Tabla 4.5 Reglas esquemáticas para LA

4.4.1.6 Reglas semánticas para LA

La semántica proporciona el significado de los elementos de LA, la fig. 4.6 muestra las reglas semánticas para LA, indicando el significado de los operadores, con los cuales se dará una interpretación a las fórmulas especificadas en LA. [8]
Las fórmulas temporales se interpretan sobre un modelo M que es una sucesión infinita de estados M: $s_0, s_1, s_2, ...s_j...$
Un modelo para LA es una estructura del siguiente tipo: [38]

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

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

cuales un modelo para LA puede ser considerado para representar una ejecución del modelo multiagente. Un modelo:

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

para una LA representa una corrida del Sistema MultiAgente:

$$\text{MAS} \overset{\text{def}}{=} \{ < \beta^{\circ}_i, A_i, M_i, \eta_i > : i \in \mathbf{Ag} \}$$

Es decir, tomando el MAS a partir de su estado inicial y considerando sus cambios de estado, obtenemos el modelo M, si se cumplen las siguientes condiciones:
$\forall u \in \mathbb{N} \ \forall i \in Ag \ \ bel(\sigma, i, u) = \beta^u_i \wedge action(\sigma, i, u) = A_i(\beta^u_i) \wedge sent(\sigma, u) = \cup_{j \in Ag} Mj(\beta^u_j)$
La relación de satisfacibilidad ($\models$) para la LA se establece entre pares de la forma $< M, u>$ y fórmulas de LA. Dado un modelo M y una fórmula temporal ϕ se presenta la noción de que ϕ se cumple en una posición $u \geq 0$ de M y lo que denotamos por : $< M, u> \models \phi$
Los conectivos utilizados son :
$\neg$ (not) $\wedge$ (and) $\vee$ (or) $\Leftrightarrow$ (iff) $\Rightarrow$ (if...then)

Para dar representación y una interpretación a las expresiones del lenguaje LA y para hacer una posible verificación de la especificación de un Sistema MultiAgente , se utilizan las reglas semánticas que se presentan en la fig. 4.7

$< M, u> \models$ true
$< M, u> \models (Bel\ i, \phi)$ iff $\phi \in bel(\sigma, I(i), u)$
$< M, u> \models (Do\ i, \alpha)$ iff $action(\sigma, I(i), u) = I(\alpha)$
$< M, u> \models (Send\ i, j, \phi)$ iff $< I(i), I(j), \phi > \in sent(\sigma, u)$
$< M, u> \models \neg\phi$ iff $< M, u> \not\models \phi$
$< M, u> \models \phi \vee \psi$ iff $< M, u> \models \phi$ or $< M, u> \models \psi$
$< M, u> \models O\phi$ iff $< M, u+1> \models \phi$
$< M, u> \models \otimes\phi$ iff $u>0$ and $< M, u-1> \models \phi$
$< M, u> \models \Box\phi$ iff $\forall k \geq u \ \ < M, k> \models \phi$
$< M, u> \models \Diamond\phi$ iff $\exists k \geq u$ s.t $< M, k> \models \phi$
$< M, u> \models \phi \mu\psi$ iff $\exists k \in N$ s.t. $k \geq u$ y $< M, k> \models \psi$ and
$\forall w \in N$ s.t $u \leq w < k, < M, w> \models \phi$
$< M, u> \models \phi\omega\psi$ iff $< M, u> \models \phi \mu\psi$ ó $< M, u> \models \Box\phi$
$< M, u> \models \upsilon\phi$ iff $\forall 0 \leq k \leq u \ \ < M, k> \models \phi$
$< M, u> \models \blacklozenge\phi$ iff $\exists 0 \leq k \leq u \ \ < M, k> \models \phi$
$< M, u> \models \alpha\phi$ iff $< M, 0> \models \phi$ ó $< M, u> \models \otimes\phi$
$< M, u> \models \phi\beta\psi$ iff $< M, u> \models \phi\xi\psi$ ó $< M, u> \models \upsilon\phi$
$< M, u> \models \phi \xi\psi$ iff $\exists 0 \leq k \leq u$ s.t $< M, k> \models \psi$ and
$\forall w \in N$ s.t $k < w \leq u \ \ < M, w> \models \phi$

Fig. 4.7 Reglas semánticas para LA

A continuación se da una interpretación gráfica de algunas de las reglas semánticas de la Fig. 4.7:
La regla esquemática para el operador "hasta que" ($\phi \mu \psi$), está dada por la siguiente expresión semántica:

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

Se representa en forma gráfica, en la fig. 4.8:

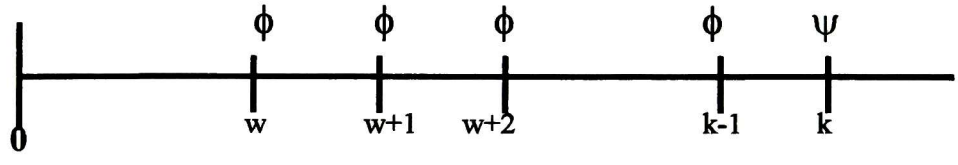


Fig. 4.8 Representación gráfica de "hasta que"

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

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

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

Tiene una representación gráfica dada por la Fig. 4.9:

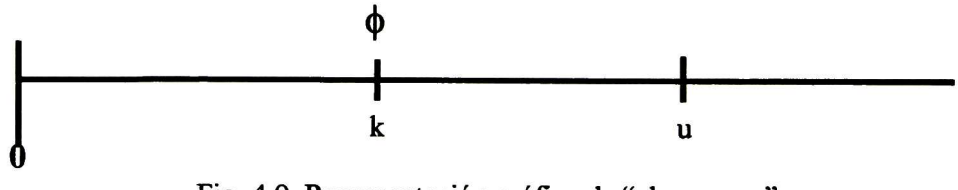


Fig. 4.9 Representación gráfica de "alguna vez"

En su interpretación dice que $\blacklozenge \phi$ es satisfecha si ϕ se cumplió en algún instante anterior o igual a u.

La regla esquemática "desde que" ($\phi\, \xi \psi$) dada por la expresión semántica:

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

Tiene la representación gráfica que ilustra la fig. 4.10

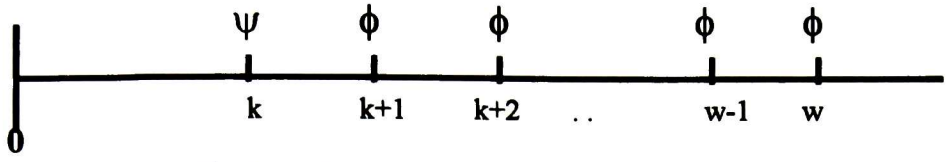


Fig. 4.10 Representación gráfica de "desde que"

En su interpretación dice que $\phi \xi \psi$ es satisfecha si ϕ se cumple en todo momento desde que ψ sea satisfecha. En esta regla ψ debe ser satisfecha eventualmente. En otras palabras, primero debe cumplirse ψ y después ϕ.

Una vez desarrollada la lógica, la cual puede ser usada para representar las propiedades de los Sistemas MultiAgente, se consideran las formas para su utilización, hay dos formas en que puede usarse la lógica: para *especificación* y para *verificación*.

4.4.1.7 Especificación

Especificación: la especificación de un sistema es una que descripción de las propiedades que se pretende que el sistema debe exhibir. Una especificación formal es aquella que es expresada (en su mayor parte) en un lenguaje matemático, como consecuencia se puede usar LA como un lenguaje de especificación formal para Sistemas MultiAgente. Esto es, nuestro lenguaje tiene contemplada esta etapa, de aquí su innovación y potencial. Para especificar las propiedades deseables en un Sistema MultiAgente se definen las siguientes clases de formulas, estas formulas se presentan de acuerdo a su jerarquía expresiva.

$\Box p$	Es una fórmula canónica de seguridad
$\Diamond p$	Es una fórmula canónica de garantía (Liveness)
$_{i=1}\wedge^{n} [\Box p_i \vee \Diamond q_i]$	Fórmula canónica de obligación
$\Box \Diamond p$	Formula canónica de respuesta
$\Diamond \Box p$	Formula canónica de persistencia
$_{i=1}\wedge^{n} [\Box\Diamond p_i \vee \Diamond \Box q_i]$	Fórmula canónica de reactividad

En donde p_i y q_i son formulas del pasado. Existen otras propiedades requeridas en la especificación de Sistemas MultiAgente. Estas propiedades tienen el propósito de garantizar la terminación apropiada del sistema.

Propiedad de corrección parcial

La propiedad de corrección parcial requiere que cada ϕ- computación final termine en un ψ-estado, por lo que no se da ninguna garantía de terminación, pero si una ϕ- computación termina, se requiere que termine en ψ- estado; es decir, en un estado que satisfaga la postcondición ψ. La corrección parcial puede expresarse por la siguiente fórmula:

$$\phi \rightarrow \Box(\text{ at-L} \rightarrow \psi)$$

En donde L es el conjunto de localidades (l_1, l_2 ... l_k). Esta formula indica que si ϕ ocurre inicialmente, entonces en donde quiera que el programa alcance el conjunto terminal L, ocurre ψ.

Propiedad de corrección total

La propiedad de corrección total requiere que cada ϕ- computación termine en un ψ- estado, entonces la corrección total garantiza la terminación en un buen ψ- estado. Esta propiedad se especifica por la siguiente fórmula:

$$\phi \rightarrow \Diamond (\text{at-L} \wedge \psi)$$

Esta fórmula establece que cada estado inicial ϕ es seguido por ψ- estado final. Esta es una propiedad de garantía (liveness).

Propiedad de Terminación

La propiedad de terminación requiere que cada ϕ- computación termine. Esto puede ser especificado por la formula:

$$\phi \rightarrow \Diamond \text{at-L}$$

Esta formula establece que cada ϕ- estado inicial es seguido por un estado terminal. Esta fórmula es considerada como una fórmula de garantía.

En este trabajo de tesis existen dos propiedades primordiales que se desea especificar en un los Sistemas MultiAgente:

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

Para describir estas propiedades se aplica la lógica LA, especificando de manera formal el protocolo de comunicación basado en el speech-Act; con el propósito de alcanzar la parte del entendimiento en el proceso de comunicación entre agentes, bajo un esquema de actividad cooperativa.
Tomar como ejemplo un protocolo simple para la actividad cooperativa, en este protocolo los agentes se comunican por medio de mensajes, en este caso tres:

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

Se asumen dos predicados de dominios

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

Un agente que recibe un Request de un agente amigo para hacer alguna acción, la hará eventualmente, pero no lo hará de otro modo (las acciones solo serán hechas por la petición de

un agente amigo). Si un agente recibe un mensaje Inform, este agregará el hecho a las creencias, solo si el emisor está acreditado.
Este protocolo simple puede especificarse con facilidad, primero consideremos que un agente que recibe un Request es un agente amigo, entonces hará una acción, por la propiedad de vivacidad. Se especifica de la siguiente manera:

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

Siempre se cumple que si *i* envía a *j* un Request para α y *j* cree que *i* es amigo, entonces eventualmente se ejecuta α
Lo siguiente expresa la propiedad de seguridad, es decir que si un agente hace alguna acción entonces la ejecución de la acción debe estar precedida por un Request por parte del agente amigo, para ejecutar la acción.

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

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

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

El modelo se generaliza tomando las acciones α como parte del conjunto Ac (constantes de Acción), para todas las acciones expresadas en los mensajes considerados en el Speech-Act: [50].

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

4.4.2 El lenguaje objeto (LCIASA)

Esta sección tiene por objetivo presentar el lenguaje objeto, cual los agentes puedan establecer un diálogo o interacción (LCIASA), El lenguaje LCIASA es un lenguaje para el desarrollo de aplicaciones multiAgente es decir aplicaciones cooperativas. LCIASA es un lenguaje de comunicación entre agentes (ACL), esta clasificado como un lenguaje de comunicación basado en el modelo de comunicación humana (speech act).

En este trabajo de tesis se utiliza LCIASA para dar una interpretación computaciónal al modelo de agentes propuesto por AML, proporcionando un lenguaje de programación con una estructura sintáctica. Con el fin de hacer la especificación del SMA de una manera mas sencilla y legible. Ya que las sentencias de LCIASA están expresadas en un lenguaje cotidiano,"comprensible, evidente para que a una persona con escasos conocimientos de la lógica y de la computación se le facilite la comprensión e instrumentación de aplicaciones de sistemas multiAgente

Las sentencias de LCIASA se consideran como acciones compuestas de los mensajes de KQML, con un objetivo predefinido, expresadas en un lenguaje coloquial y con la finalidad de que el usuario pueda expresar sus necesidades de acceso a la información de una manera simple y adecuada a la forma de expresarse cotidianamente. Asimismo con las acciones compuestas de LCIASA es posible autenticar cada una de las acciones involucradas en el proceso de la interacción entre agentes, con el fin de garantizar las propiedades de seguridad y vivacidad del sistema especificado
LCIASA corresponde al lenguaje Objeto de agentes. El lenguaje LCIASA consta de un conjunto de sentencias basado en las acciones atómicas de KQML[34] (speech-act). La fig. 4.11 muestra la dependencia entre el LCIASA y KQML

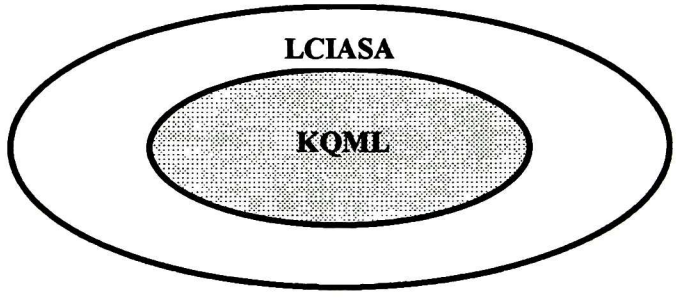


Fig. 4.11 Subordinación de LCIASA respecto a KQML

Con el conjunto de mensajes se tiene acceso a la información almacenada en un grupo de dispositivos interconectados bajo un esquema distribuido abierto, en donde se da una interacción entre agentes para acceder a la información independientemente del contexto, tecnología o plataforma empleada en cada uno de los componentes del sistema.

4.4.2.1 Sintaxis de LCIASA

La sintaxis es considerada como el conjunto de reglas que generan las sentencias escritas correctamente en el lenguaje, incluyendo particularidades como la ubicación de las palabras y los signos de puntuación. Para especificar la sintaxis de LCIASA, en este trabajo se utiliza la notación BNF.
La fig. 4.12 muestra la sintaxis para un mensaje de LCIASA. Este mensaje corresponde a la sintaxis del mensaje de KQML y este es considerado como una acción primitiva de LCIASA.

```
<message>::=(<word>{<whitespace>:<word>
 <whitespace > <expression>}*)
<expression>::=  <word>| <quotation>| <string>!
          ( <word>{<whitespace> <expression>})
<word>::=<character><character>*
<character>::=<alphabetic>|<numeric>|<special>
<special>::=<|>|=|+|-|*|/|&|^|~|_|@|$|%|:|.|?|!
```

<quotation>::='<expression>|'<comma- expression>
<comma-expression>::=>::= <word>| <quotation>| <string>!,<coma-expression(<word>
{<whitespace> <coma-expression>}*)
<string>::="<stringchar>*"| #<digit><digit><ascii>*
<stringchar>|::=\<qascii>|<ascii>-\-<double-quote>

Fig.4.12 KQML sintaxis del mensaje

Tomando en cuenta la función que los "mensajes" llevan a cabo, estos se agrupan en los siguientes tipos:

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

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

<mensaje>::=(<inf-act-stat>|<BD-act-stat>|<Asn-act-stat>|<Req-act-stat>|<Mresp-act-stat>
|<Def.cap-act-stat>|<Notif-act-stat>|<Conect-act-stat> |<Fac-act-stat>)
< inf-act-stat> ::= Tell|Deny|Untell|
<BD-act-stat> ::= Insert|Delete|Delete_One|Delete_All
< Asn-act-stat> ::=Error|Sorry
<Req-act-stat> ::=Evaluate|Reply|Ask-if|Ask-about|Ask-one
|Ask-all
<Mresp-act-stat> ::= Stream-about|Stram-all|Eos
<Def.cap-act-stat> ::= Advertise
<Notif-act-stat> ::= Subscribe|Monitor
<Conect-act-stat> ::= Register|Unregister| Forward |Broadcast
|Pipe|Break|Transport_Adress
<Fac-act-stat> ::= Broker-one|Broker-all|Recommend-one
|Recommend-all|Recruit-one|Recruit-all

En la tabla 4.7 se muestra el conjunto de mensajes para KQML con su significado y función asociado a cada uno de ellos.

Conjunto de Mensajes de KQML

Nombre	Significado
Archieve	S quiere que R haga algo verdadero en su medio ambiente
Advertise	S esta particularmente disponible para procesar un "mensaje"
ask-about	S quiere todas las sentencias relevantes de la VKB[2] de R
ask-all	S quiere las respuestas de todos los R`s a una pregunta
ask-if	S quiere saber si la sentencia está en la VKB de los R´s

ask-one	S quiere una de las respuestas de los R`s a una pregunta
Break	S quiere que R rompa una línea establecida
Broadcast	S quiere que R envíe un "mensaje" a todas las conexiones
broker-all	S quiere que R colecte todas las respuestas a un mensaje
broker-one	S quiere que R obtenga ayuda en respuesta a un mensaje
Deny	El mensaje contenido no se aplica a S (nunca mas)
Delete	S quiere que R elimine una sentencia de su VKB
delete-all	S quiere que R elimine todas las sentencias que coincidan de su VKB
delete-one	S quiere que R elimine una sentencia que coincida de su VKB
Discard	S no quiere que los R`s mantengan respuestas a mensajes previos
Eos	Fin de una corriente de respuestas a una pregunta anterior
Error	S considera que el mensajes anterior de los R`s está mal formado
Evaluate	S quieres que R simplifique la sentencia
Forward	S quiere que R enrrute el mensaje
Generator	Lo mismo que un estandby de una corriente completa
Insert	S pide a R que agregue el contenido a su VKB
Monitor	S quiere actualizar las respuestas de los R`s a una corriente total
Next	S quiere la siguiente respuesta de R´s a un mensaje previo
Pipe	S quiere que R enrrute todos los mensajes a otro agente
R[1]eady	S esta listo para responder a los mensajes previos de R`s
Recommend-all	S quiere los nombres de todos los agentes que pueden responder un mensaje
Recommend-one	S quiere el nombre de un agente que pueden responder un mensaje
Recruit-all	S quiere que R obtenga todos los agentes disponibles para responder a un mensaje
Recruit-one	S quiere que R obtenga otro agente responda a un mensaje
Register	S puede entregar mensajes a un agentes designado
Reply	Comunica una respuesta esperada
Rest	S quiere que R´s proporcionen las respuestas a un mensaje previo
Sorry	S no puede proporcionar una respuesta mas detallada
Standby	S quiere que R este listo para responder a un mensaje
stream-about	Versión de Respuesta múltiple para ask-about
Stream-all	Versión de respuesta múltiple para ask-all
Subscribe	S quiere actualizar las respuestas de R`s a un mensaje
Tell	La sentencia está en la VKB de S`s
Transport-address	S asocia un nombre simbólico con una dirección de transporte
Unregister	Un deny de un register
Untell	La sentencia no está en la VKB de S`s

Tabla 4.6 Conjunto de mensajes Para el Emisor S y el receptor R

[1] VKB Base Virtual de Conocimiento

Como se ilustró anteriormente en la fig.4.11 el LCIASA toma como base los mensajes de KQML para formar las sentencias compuestas que forman al lenguaje. La tabla 4.8 muestra los mensajes del lenguaje LCIASA con su función asociada.

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

Tabla 4.7 Conjunto de mensajes para LCIASA

Los mensajes de LCIASA se agrupan de acuerdo a las acciones que los agentes realizan cuando estos mensajes son interpretados y ejecutados. Estos grupos funcionales para LCIASA son: acciones de agentes, autorización, cifrado, autentificación, verificación y operaciones en la VKB. A continuación se muestran los grupos de acciones, haciendo una breve descripción de sus correspondientes funciones:

- **Acciones de agentes**
  Acciones que realiza el agente para responder a una petición hecha por otros agentes

  Encuentra-resuelve-mensaje, Localidades-de-información,
  Localiza-todas-soluciones, Responde-con-la-solución.

- **Acciones de autorización**
  Acciones de los agentes para confirmar o denegar aciones posteriores

  Visualiza-VKB, Confirma-firma-digital, Valida-agente.

- **Acciones de cifrado**
  Mecanismos de protección a la revelación de información

  Dispone-mensaje
  Protege-mensaje

- **Acciones de Autentificación**
  Confirmación de la legitimidad de los agentes y mensajes

  Autentifica-agente
  Autentifica-emisor
  Autentifica-mensaje
  Autentifica receptor

- **Acciones de Verificación**
  Acciones para comprobar la consistencia de los planes

  Comprueba-plan
  Verifica-orden-mensaje

- **Acciones de operación en la VKB**
  Acciones que permiten administrar la base de conocimientos

  Admite-creencia
  Elimina-plan
  Incorpora-VKB
  Omite-creencia

EL mecanismo de interacción entre los agentes se lleva a cabo por medio de un intercambio de mensajes, en donde estos mensajes son agrupados para constituir un plan cuyo propósito es el de realizar una cierta tarea. La sintaxis de LCIASA está basada en el concepto de plan. La fig. 4.13 ilustra en forma gráfica la estructura sintáctica general de LCIASA.

En la fig. 4.13 se observa que cada plan está caracterizado por un identificador, un bloque del plan que contiene los atributos y acciones del plan: las precondiciones, el cuerpo del plan y las postcondicones.

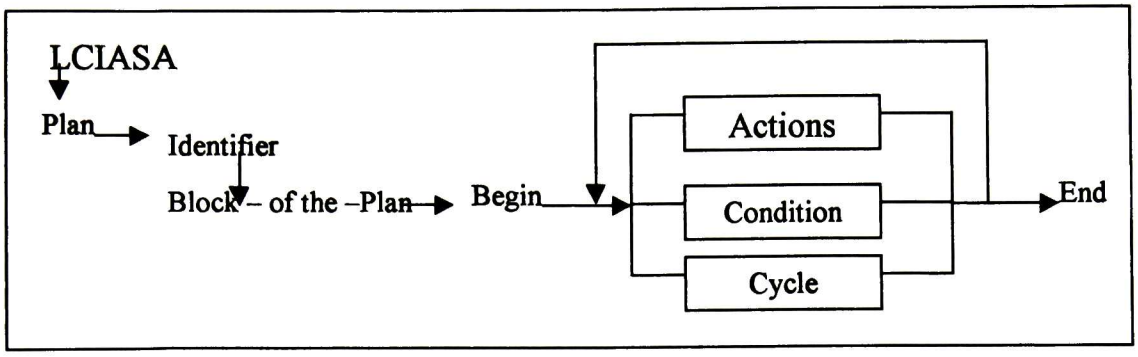


Fig. 4.13 Estructura general para la sintaxis de LCIASA

A continuación, en la Fig. 4.14 se expresa la sintaxis de LCIASA, utilizando la notación BNF para este propósito.

<LCIASA> ::= <Plan>
<Plan> :: = <Heading - of the - Plan> Begin <Block - of the - Plan > End
<Heading - of the - Plan> ::= Plan <Identifier>
<Block -of the - Plan ::= Begin - Plan [<Pre-conditions >] [<Declaration- of-Variables>]
<Body - of the - plan > [<Post-conditions >] End - Plan

<Pre-conditions> ::= <Attributes - of the - Plan> <Conditions - of - Invocation >
<Attributes - of the - Plan> ::=Available | Reserved | Activated | Disabled
<Condition - of - Invocation> ::= <Condition - of - Activation> | <Event - of - activation>
<Condition - of - Activation ::= <Action > | <Expression >
<Event - of - activation ::= <Action> | <Expression >

<Declaration - of - Variables> ::=Variables < Lists - of - Identifiers > : <Type >
<Lists - of - Identifiers> ::= <Identifier > {, <Identifier > }
<Type> ::= <Integer > | <Agent > | <Real > | <Boolean >
<Integer> ::= <Digit > {<Digit >}
<Agent> ::= <Id- Agent >
<Id- Agent ::= <Identifier>
<Real> ::= <Integer > . <I Integer>
<Boolean> ::=True | False

<Body - of the - plan ::={<Actions> }
<Actions> ::= <Action - AML > | <Action - Ordinary >
<Action - AML ::= Send (<Agent > <Agent > <Action - Simple >
| Bel (<Agent> <Expression - Boolean >) | Request(<Action - Simple >)
| Do (<Agent> , <Action - Simple>) | Inform(<Action - Simple >)
<Action - Ordinary> ::= <Action - Simple > | <Action - Structured >

<Action - Simple> ::= <Action - Primitive - KQML> | <Action - Multiple >
<Action - Primitive - KQML> ::= <**message** >
<Action - Multiple> ::= <**Actions - Compound**>

<**message**> :: = <Informative> | <Base - of - Data > | <Responses> |<Notification> |
<Connectivity <| <Definition - of - Capacities <| <Facilities> | <Query>
<Of - Effect < | <Multi-response> | <Generating>
<Informative> ::= Tell | Deny | Untell
<Base - of - Data> ::= Insert | Delete | Delete - One | Delete - All
<Answer > ::= Error | Sorry
<Query> ::= Evaluate | Reply | Ask-if | Ask-about | Ask-one | Ask-all
<Multi-response> ::= Stream-about | Stream-all | Eos
<Definition - of - Capacities> :: = Advertise
<Notification> :: = Subscribe | Monitor
<Connectivity> ::= Register | Unregister | Forward | Broadcast | Pipe | Break
| Transport-Address
<Facilities> ::= Broker-one | Broker-all | Recommend-one | Recommend-all | Recruit-one
| Recruit-all
<Of - Effect > ::= Achieve | Unachieve
<Generating> :: = Stand by | Ready | Next | Rest | Discard | Generator

<**Actions – Compound**> ::= <Actions - of - Agent>| <Authorization> |<Encoded >
| <Authentication <| <Verification <| <Operations - in - the - VKB>
<Actions - of - Agent ::= Finds - Solve - Message | Locations - of - Information
| Locate - all - Solutions | Respond - with - the - Solutions | Analyze - Plan
<Authorization> ::= Visualize - VKB | Confirm - Signs - Digital | Validate - Agent
<Encoded> ::= Dispose - Message | Protects - Message
<Authentication> ::= Authenticates - Agent | Authenticates - Originator | Authenticates -
Message | authenticates - Receiver
<Verification> ::= Checks - Plan | Verifies - Order - of - Message
<Operations - in - the - VKB> ::= Admits - Belief | Eliminates - Plan | VKB Incorporates
| Omits - Belief | Add - Plan

<Action - Structured> ::= <Action - Conditional >| <Action - Repetitive>
<Action - Conditional> ::=If <Expression > Then <Actions> Else <Actions>
<Action - Repetitive> ::= While <Expression> Do <Actions>
<Expression>::= <Identifier > <operator> <Identifier >

<Post-conditions> ::= <Conditions - of - abort >| <Action - of Failure>
| <Action - of - Approval >
<Condition - of - abort> ::= <Action - Simple>| <Expression >
<Action - of Failure>::= <Action - Simple > | <Expression >
<Action - of - Approval> ::= <Action - Simple >| <Expression>

<Identifier>::= <Letter> {<Letter>| <Digit >}

<Letter> ::=A | B |... | Y| Z | a | b |... | y | z
<Digit> ::=1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
<Operator-arithmetic> ::= + | - | * | /
<Operator - Boolean ::= ∨ | ∧ | ¬
<Operator - Relational ::= ≤ | ≡ | ≥ | ≧ | ≦

<Integer> ::= <Digit>{<Digit >}
<Expression - Boolean>::= <Action - Simple>[{<Operator- Boolean ><Action - simple> }] ≡
<Boolean>
<Agent> ::= <Id- Agent>
<Id- Agent ::= <Identifier>

Fig. 4.14 Sintaxis de LCIASA

En la especificación de la sintaxis para LCIASA se percibe la forma en que LCIASA está asociado con AML. Esta asociación se detecta observando primero. Que las acciones de AML corresponden a los mensajes de LCIASA, segundo: que LCIASA proporciona una estructura sintáctica a AML.

4.4.2.2 Semántica de LCIASA

La semántica relaciona al lenguaje con situaciones o con la representación de situaciones en el mundo. Con la semántica se obtiene una interpretación de las acciones ejecutadas durante el intercambio de información entre agentes.

La semántica del LCIASA está dada por un contexto en donde cada agente observa las capacidades de cada uno de los otros. Cada agente es visualizado como si administrara una base de conocimiento. Es decir, la comunicación con el agente se hace con respecto a su base de conocimiento.

La implementación del agente no es necesariamente estructurada como una base de conocimiento. Es decir se puede utilizar un simple sistema de bases de datos o un programa que utilice una estructura de datos especial, este programa debe trasladar el código empaquetado a una abstracción basada en conocimiento, para ser utilizada por otros agentes. Entonces se dice que cada agente maneja una base de conocimiento virtual (VKB).

Cuando se definen los "mensajes", es útil clasificar las sentencias dentro de la VKB en dos categorías: "creencias" e "intenciones". Una creencia de un agente codifica la información que se tiene del el mismo y de su medio ambiente, incluyendo la VKB de otros agentes. Una intención codifica los estados de su medio ambiente externo que el agente tratará de alcanzar. Los agentes sostienen una conversación alrededor de sus VKB de otras VKB utilizando el LCIASA, pero la codificación de las sentencias en la VKB puede utilizar una variedad de lenguajes de representación.

4.5 Protocolos de Interacción

El protocolo de Interacción se puede ver como la descripción secuencia de mensajes intercambiados y de las reglas seguidas por los agentes para lograr un objetivo específico. Estas secuencias se agrupan para formar una sucesión coherente de acciones asociadas a un plan, intención o protocolo que el agente esté capacitado para ejecutar [37]

Los protocolos de interacción gobiernan el intercambio de una serie de mensajes entre los agentes. Es decir, los protocolos de interacción controlan el diálogo que los agentes sostienen para resolver un problema en forma conjunta, exhibiendo que acciones se llevan a cabo y en que momento estas son ejecutadas.

Los protocolos de interacción abordados en este trabajo son:

- Protocolos de Coordinación: Los cuales nos permiten lograr la coherencia en la solución de SMA. En este trabajo se utilizan los mecanismos de mercadeo basados en contratos.

- Protocolos de Cooperación: definen los mecanismos que soportan la actitud de los agentes que deciden trabajar juntos para resolver una tarea. Serán utilizados protocolos basados en contratos.

- Protocolos de negociación: Definen los elementos con los cuales sea posible que un conjunto de agentes llegue a un estado mutuamente aceptable. Se utilizará el lenguaje LCIASA como medio para que los agentes logren intercambiar mensajes.

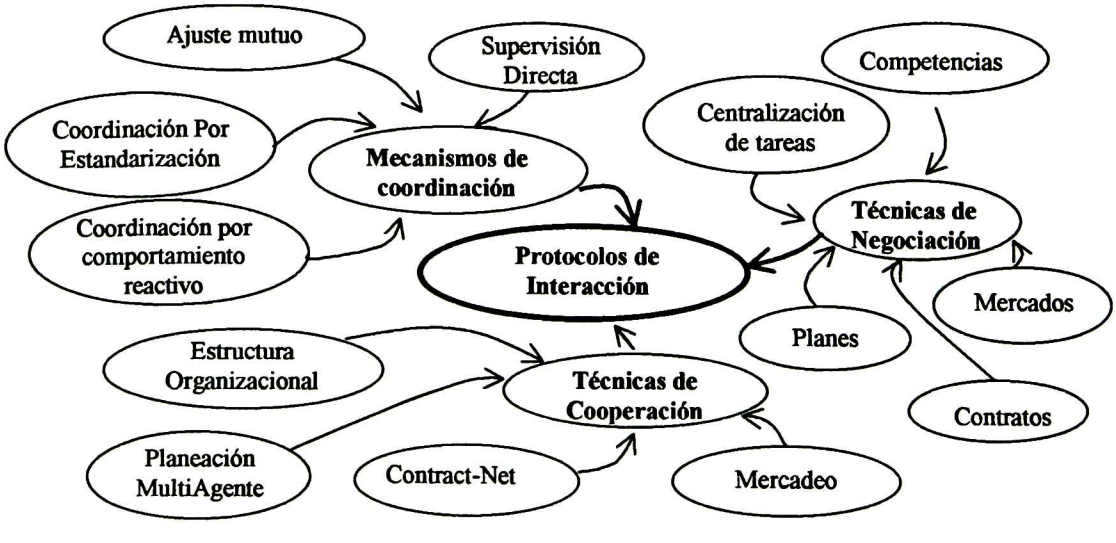


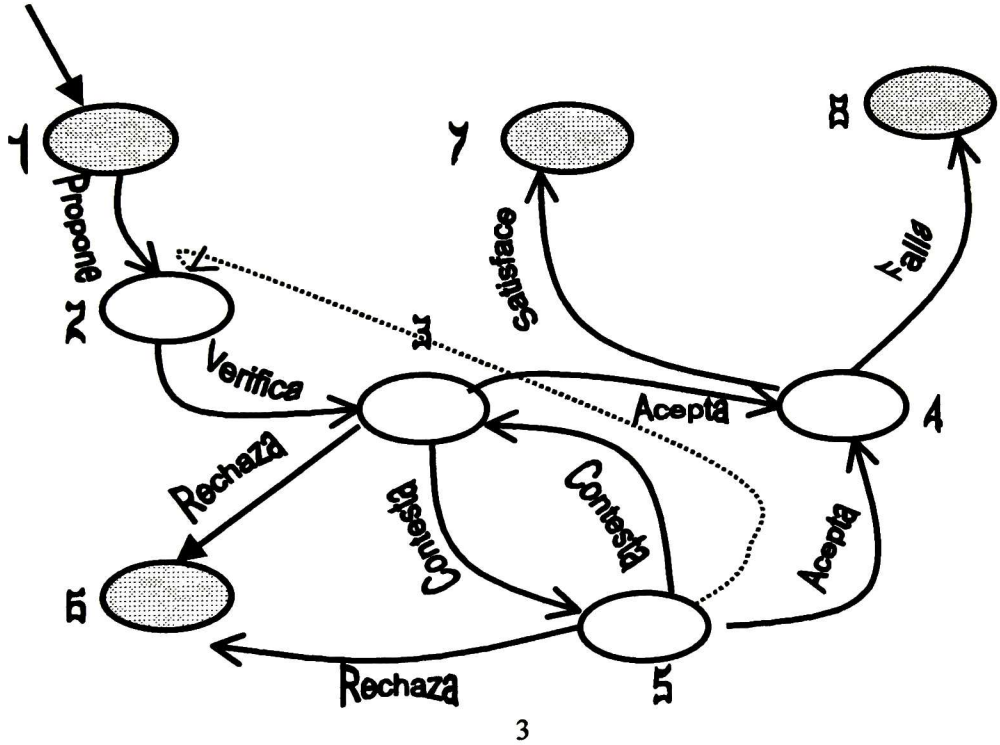
Fig.4.15 Esquema de los protocolos de Interacción

En la fig. 4.15 se muestra un esquema que agrupa algunas de las técnicas y mecanismos sugeridas para definición de los protocolos de interacción.

En este trabajo de tesis se crea un protocolo orientado a las actividades de comercio electrónico con el cual definiremos el proceso de interacción entre los agentes. Este protocolo es llamado "Contract Bussiness" (CB) y será descrito posteriormente en el capítulo 5.

A continuación se ilustra el esquema general que representa la interacción entre los agentes. Este esquema general en una guía para definir los protocolos básicos y algunos protocolos de interés requeridos en el proceso de entendimiento entre los agentes.

La figura 4.16 exhibe la posibilidad de especificar los mensajes factibles que se presentan en la interacción entre agentes, asimismo, definir las reglas que permitan a los agentes llevar a cabo sus funciones mediante un intercambio de mensajes, ésta secuencia ordenada de mensajes es conceptuado como un protocolo de interacción para llevar a cabo una función o tarea específica.



3

Fig. 4.16 Esquema general de transiciones para la interacción de agentes

Con el esquema general de transiciones para la interacción de agentes, es posible especificar todas y cada una de las sentencias de LCIASA como secuencias de las acciones primitivas de KQML, o como series de las sentencias del propio LCIASA

La Fig. 4.17 muestra la secuencia de acciones que efectúa del agente A para encontrar al agente(en este caso el agente B) que este habilitado para responder a una solicitud del agente A y después obtener la información requerida de la VKB del agente B.

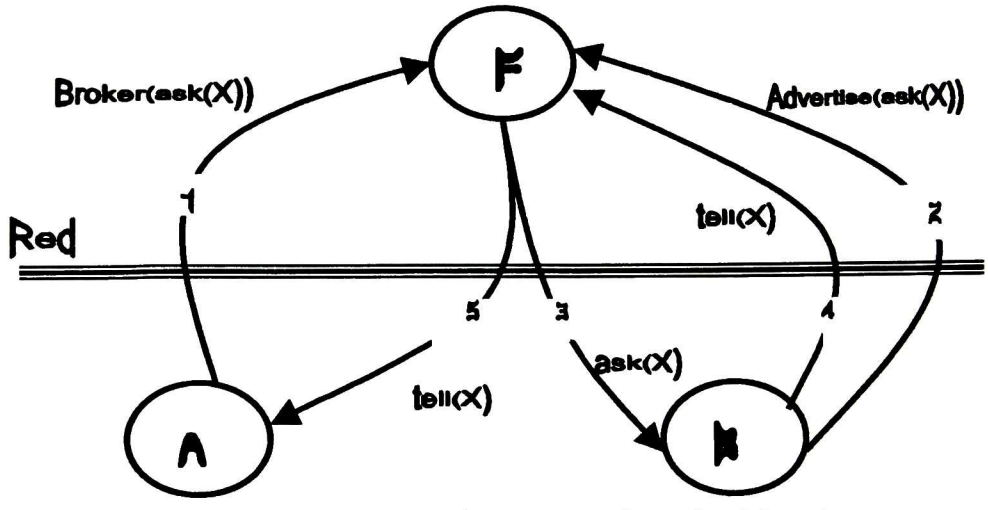


Fig. 4.17 Protocolo Encuentra-Resuelve-Mensaje

En la Fig. 4.17 se muestra la secuencia de acciones, las cuales corresponden a la sentencia de LCIASA Encuentra-Resuelve-Mensaje. Aquí se hace uso de un facilitador (F), el cual tiene la función de servir como una interfaz común entre los agentes. De la misma manera a través del intercambio de mensajes se logra definir el conjunto de protocolos para LCIASA y otros que sean de interés a la aplicación que se realiza.

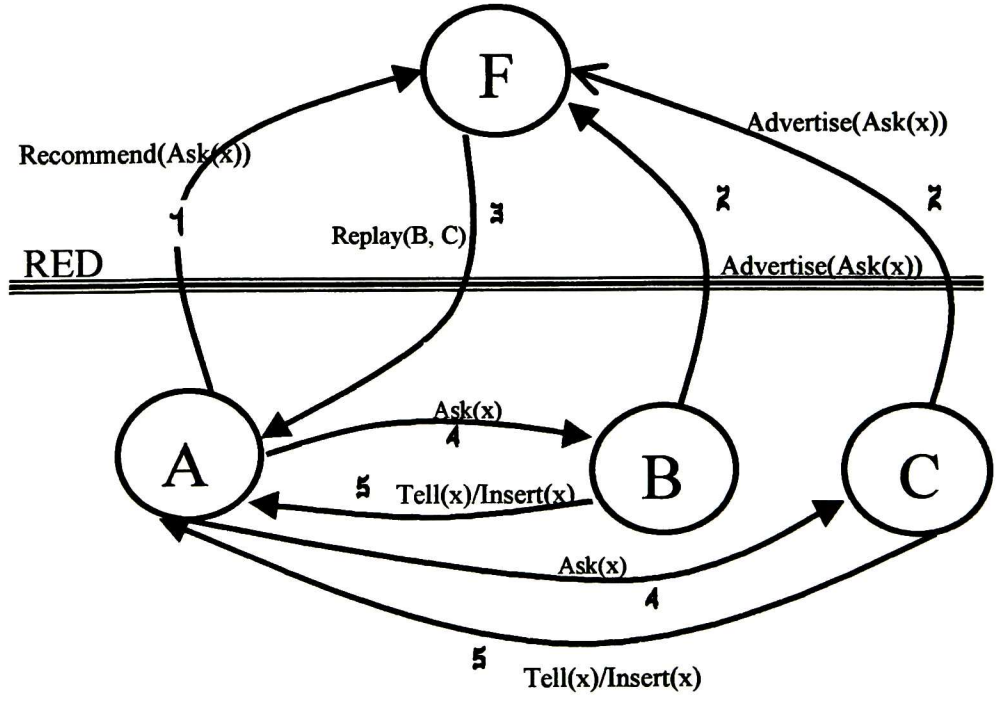


Fig. 4.18 Lectura/Escritura en archivos compartidos

En la fig. 4.18 se presenta un esquema para un Sistema MultiAgente que permita la lectura/escritura de información en archivos compartidos. Aquí también se utiliza un facilitador (F) como una interfaz común para los agentes.

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

Conclusión

En éste capítulo se presenta un enfoque original e innovador para identificar, asociar y organizar los agentes en un modelo jerárquico que represente la solución del problema que se está resolviendo. También se presenta el lenguaje de comunicación de agentes LCIASA, el cual manifiesta su potencial al tener asociado el metalenguaje de agentes AML, haciendo posible una verificación formal de las acciones especificadas con este lenguaje, asimismo se proponen los protocolos de interacción cuyo propósito consiste en definir los mecanismos de coordinación, cooperación y negociación entre los agentes.

Capítulo 5

Fase de Implementación

Resumen

En este capítulo se aborda la fase de implementación, en donde se exponen los conceptos necesarios que nos permiten pasar de la especificación abstracta a un sistema computacional concreto. Primero; se estudia la arquitectura del agente [42], en la cual se definen las estructuras de datos y la forma en que estas estructuras están organizadas para describir las acciones dentro del sistema. Segundo, se concreta el protocolo CB: "Contract Business" [13]; para construir este protocolo, la fase de implementación toma como entrada los modelos y conceptos definidos en la fase de especificación, estos conceptos son seleccionados para obtener el protocolo que represente la solución dinámica del problema que se está resolviendo. Una vez construido el protocolo CB, se utiliza el lenguaje LCIASA [8] para producir una codificación de las acciones representadas por el protocolo. Y finalmente se hace referencia a la plataforma de agentes desarrollada como parte de este trabajo de tesis por Zulma S. Aguirre P. [1], en la cual se exhibe la ejecución a la solución del problema a resolver.

5.1 Introducción

La fase de implementación estudia los conceptos, modelos y procedimientos con los cuales pasar de modelos que definen la especificación a un sistema computacional concreto. En este trabajo de tesis se proponen dos alternativas para implementar la especificación de un problema específico: ejecución directa y la plataforma de agentes. La fig. 5.1 muestra un esquema en donde se describen los conceptos asociados a éstas dos aproximaciones propuestas para la implementación del problema a resolver.

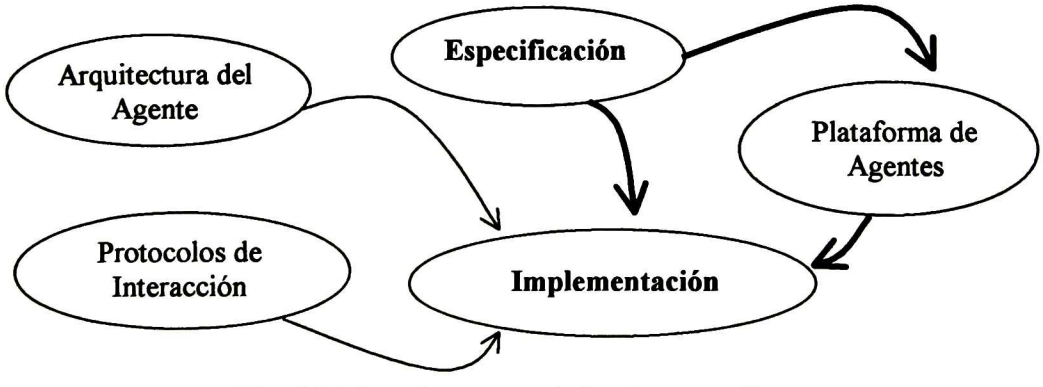


Fig. 5.1 Mecanismos para la implementación

En la Fig. 5.1 se aprecian las dos aproximaciones propuestas para cumplimentar la fase de implementación. *La primera aproximación* a la que se denomina como aproximación directa. Para satisfacer esta aproximación, primero se define una arquitectura del tipo BDI para representar la funcionalidad interna del sistema, posteriormente se describe el protocolo CB para representar la solución dinámica del problema a resolver y finalmente se utiliza LCIASA para codificar el protocolo CB. *La segunda aproximación* requiere de crear una plataforma de agentes para trasladar la especificación del problema a un código ejecutable.

5.2 Arquitectura del agente

En la arquitectura del agente se definen los mecanismos con los cuales pasar del modelo teórico a la práctica, además se consideran los componentes que representan el punto de vista interno del sistema, exhibiendo la funcionalidad del SMA, haciendo posible la construcción de Sistemas multiAgente, de tal forma que satisfagan las especificaciones dadas por los modelos teóricos.

La Arquitectura del agente corresponde al punto de vista del diseñador del sistema. Es decir la arquitectura define los componentes principales del agente como: los mecanismos de percepción, razonamiento, acción, almacenamiento de información, decisión, etc.; la forma en que estos elementos están relacionados, así como las estructuras de datos empleadas en la implantación del agente. La arquitectura del agente, también define la manera en que varias partes del agente pueden ser ensambladas de tal manera que este pueda realizar las acciones esperadas.

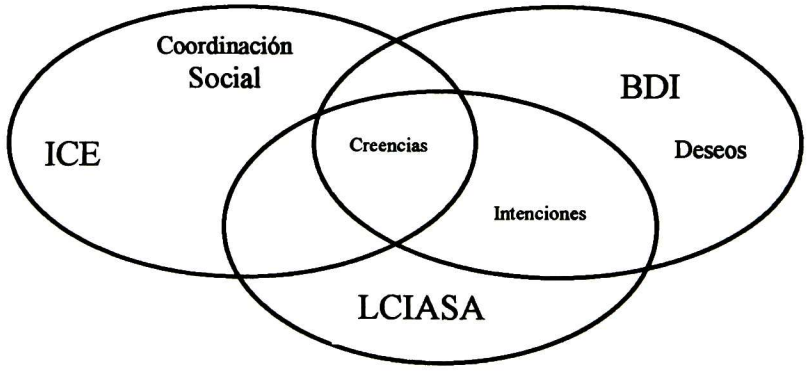


Fig. 5.2 contexto de la arquitectura para el lenguaje LCIASA

La arquitectura del agente propuesta en este trabajo, es una arquitectura BI (Belief, Intentions, por sus siglas en ingles) [9], la cual puede considerarse como un subconjunto de la arquitectura BDI [24], que a su vez son incluidas, como un caso especial en la arquitectura ICE (abreviación de $I^2C^2E^2$: Información, Intención, Comunicación, Cooperación, Evaluación, Empowerment. La Arquitectura propuesta es usada para especificar la funcionalidad de los agentes dentro del sistema. Esta arquitectura contiene las estructuras de datos y de

almacenamiento necesarias para definir las capacidades del agente. El contexto para la arquitectura del agente propuesta se ilustra en la Fig. 5.2

El tipo de arquitectura considerado para el LCIASA corresponde al tipo de arquitectura clásica llamada arquitectura deliberativa [42], que es aquella que contiene un modelo simbólico explícito del mundo, en el cual las decisiones son hechas por medio de un razonamiento lógico basado en el empate de patrones y en la manipulación simbólica.

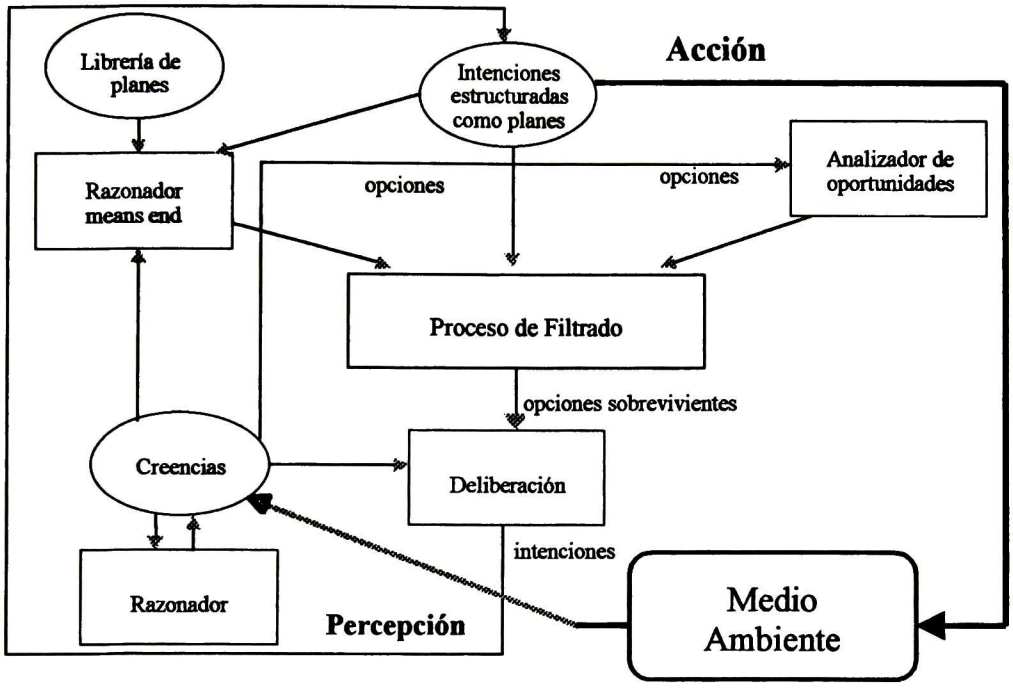


Fig. 5.3 Arquitectura del Agente

En la Fig. 5.3 se muestran las estructuras de datos clave y la forma en que estas estructuras están organizadas para describir la funcionalidad de los agentes. A continuación se hace una breve descripción de los conceptos asociados a la arquitectura del agente.

Esta arquitectura tiene su fundamento en la tradición filosófica del entendimiento del *razonamiento práctico*, que es el proceso de decidir momento a momento, cual acción ejecutar en la consecución de las metas propuestas. El razonamiento práctico involucra dos procesos importantes: primero, decidir que metas se quieren alcanzar, al que se le llama proceso de deliberación y segundo decidir como se van a lograr las metas, a este proceso se le llama razonamiento "means-end".

Una *librería de planes.* Conjunto de planes almacenados como procedimientos, esta librería puede ser vista como un subconjunto de las creencias que tiene el agente, acerca de que planes y bajo que circunstancias son requeridos para ejecutar una tarea. *Las Intenciones*

estructuradas como planes, son los planes que el agente está ejecutando en el momento presente. Adicionalmente la arquitectura del agente contiene un *razonador,* que le permite discernir acera del mundo. El razonador "means-end" se invoca una vez que se ha formado una intención, para cada uno de los planes parciales existentes, con el fin de proponer subplanes para determinar cuales planes pueden ser usados para alcanzar las intenciones del agente, un *analizador de oportunidades* que percibe el medio ambiente con el fin de determinar opciones posteriores para el agente, las opciones están dadas por la función de generación de opciones de la siguiente manera:

$$\text{Opciones: } \wp(\text{Bel}) \text{ X } \wp(\text{Int}) \rightarrow \wp(\text{Int})$$

La cual mapea un conjunto de creencias y un conjunto de intenciones a un conjunto de intenciones. Un *proceso filtrador* y un *proceso deliberador,* el proceso filtrador es el responsable de concretar el subconjunto de los cursos de acciones potenciales del agente, que tienen la propiedad de ser consistentes con las intenciones actuales del agente, la elección entre las opciones que compiten es hecha por el *proceso de deliberación* el cual recibe las opciones sobrevivientes después del filtrado y pondera las opciones unas contra otras y actualiza las intenciones estructuradas como planes. El proceso de deliberación está representado por la función de filtrado, de la siguiente manera:

$$\text{Filtro: } \wp(\text{Bel}) \text{ X } \wp(\text{Int}) \rightarrow \wp(\text{Int})$$

La cual actualiza las intenciones del agente con base a sus intenciones anteriores y a sus creencias actuales. El *Almacén de creencias* contiene una representación de lo que el agente conoce de sí mismo y de su medio ambiente, como se muestra en la Fig. 5.3

Con el propósito de ofrecer una mayor legibilidad a los conceptos y funciones correspondientes a la arquitectura del agente, como se ilustra en la Fig. 5.3, a continuación se exhibe un esquema simplificado de la arquitectura del agente, por medio de la Fig. 5.4.

En donde:

La base de conocimientos mantiene el conocimiento del agente mismo, de su medio ambiente externo y de otros agentes.

Las acciones representan los medios del agente para percibir el mundo externo, actuar sobre el mundo externo y comunicarse con otros agentes. Las acciones pueden dividirse en acciones primitivas y complejas (acciones definidas por la composición de otras acciones).

El motor define el comportamiento del agente, esto es la forma de actuar en respuesta a su conocimiento y a las interacciones con el mundo externo.

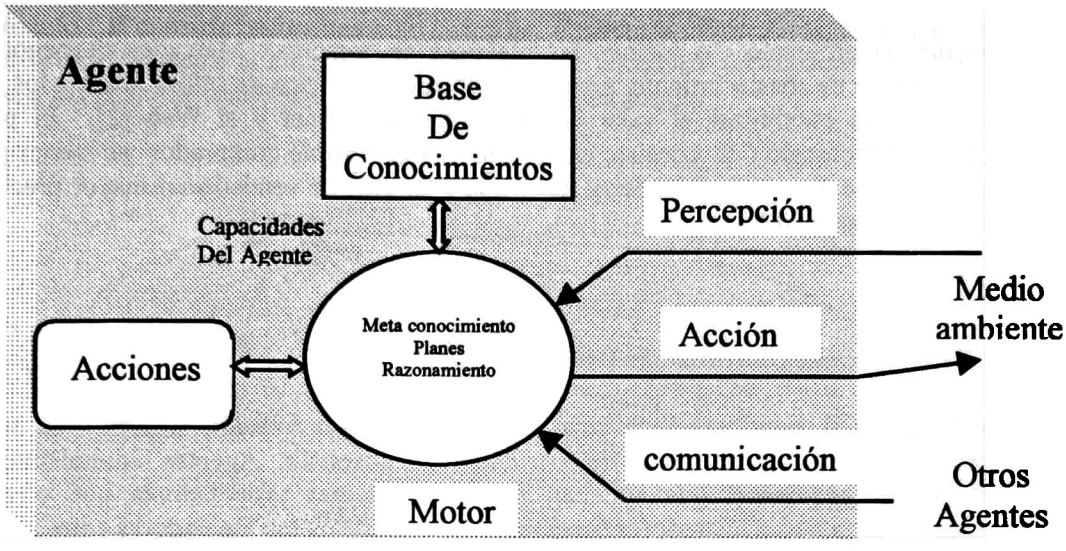


Fig. 5.4 Estructura de un agente

5.3 Descripción del protocolo CB

Como se mencionó anteriormente en la fase de implementación se toma como entrada la especificación del problema propuesto, así como los modelos definidos en el capítulo anterior para crear el protocolo CB, el cual representa la solución al problema propuesto. Una vez que se ha obtenido el protocolo, se utiliza LCIASA para codificarlo, obteniendo así la implementación del problema propuesto.

El protocolo CB Integra los procesos de negocios electrónicos (estándares OBI, Mercados centralizados) con las estructuras de coordinación y los mecanismos de cooperación y negociación requeridos para resolver adecuadamente el problema propuesto.

Para concretar el protocolo se consideran los siguientes aspectos: 1) estudios previos de los mercados centralizados de donde se toma la estructura organizaciónal que define la forma en que los agentes se agrupan para cumplir con una tarea específica [33]. Esta estructura está constituida por una comunidad de diferentes agentes: Administradores independientes de productos, agentes que requieren servicios (Agentes compradores), agentes que ofrecen sus servicios (Agentes vendedores) y agentes funcionales que actúan como mediadores entre ambos (Agentes facilitadores). 2) mecanismos de coordinación [53], en este caso se utilizan herramientas de mercadeo basados en contratos. 3) protocolos de cooperación [16], para resolver la aplicación propuesta se utiliza el protocolo "Contact net" 4) protocolos de negociación para esta aplicación los agentes utilizan el protocolo de "regateo" [3] para alcanzar una solución mutuamente aceptable por los agentes involucrados en la solución del

problema. 5) Finalmente el protocolo CB toma los procedimientos de comercio electrónico de estándares como OBI [32], [38].

Asimismo, el protocolo CB incluye las prácticas de comercio: Mercados de negocio [32], ventas, cadenas de abastecimiento, servicios a clientes, servicios de información y los recursos necesarios para maximizar el valor comercial de la Internet y el Web [2]. El propósito principal del protocolo CB consiste en presentar a un posible comprador en menor precio al que alguien este dispuesto a vender y mostrarle a un posible vendedor el mayor precio al que alguien esté dispuesto a comprar.

5.3.1 Estructura de coordinación para el protocolo CB

El protocolo CB tiene asociada una estructura de coordinación denominada mercados centralizados [19], en este tipo de estructura se observan: Agentes administradores de productos independientes que solicitan servicios, agentes procesadores que ofrecen sus servicios y agentes administradores funcionales que actúan como mediadores entre ambos.

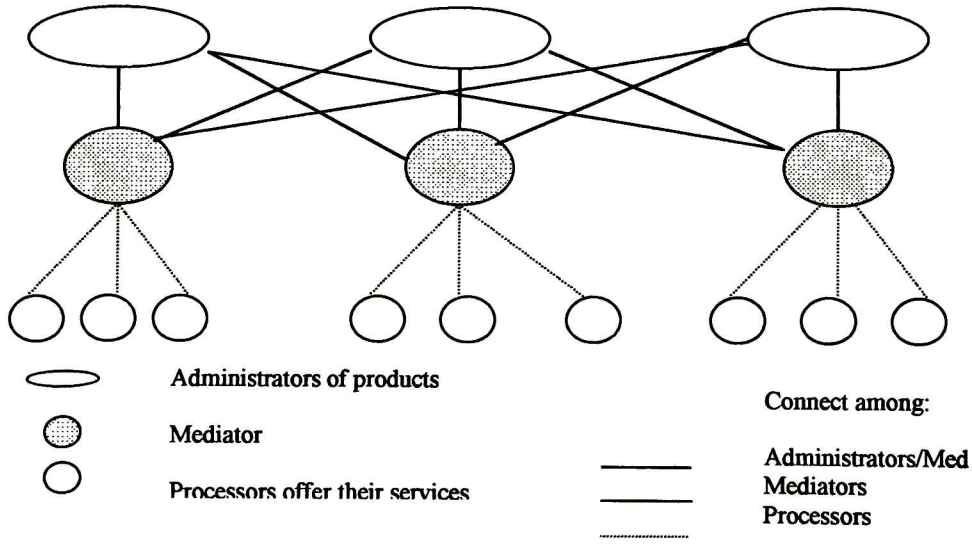


Fig. 5.5 Estructura de coordinación de mercados centralizados

En esta estructura de coordinación los agentes administradores de productos requieren conexiones hacia cada agente mediador; cada agente mediador requiere conexiones hacia todos los agentes administradores y cada agente mediador requiere conexiones hacia todos los agentes procesadores asociados a este. Como se ilustra en la Fig. 5.5.

5.3.2 Funcionalidad del protocolo CB

El protocolo CB consiste básicamente de ciclos de: Encuentra proveedores, Acuerda términos, proveer orden y soporte al cliente. En el proceso de encuentra proveedores, el comprador informa a los posibles vendedores del producto que desea adquirir.

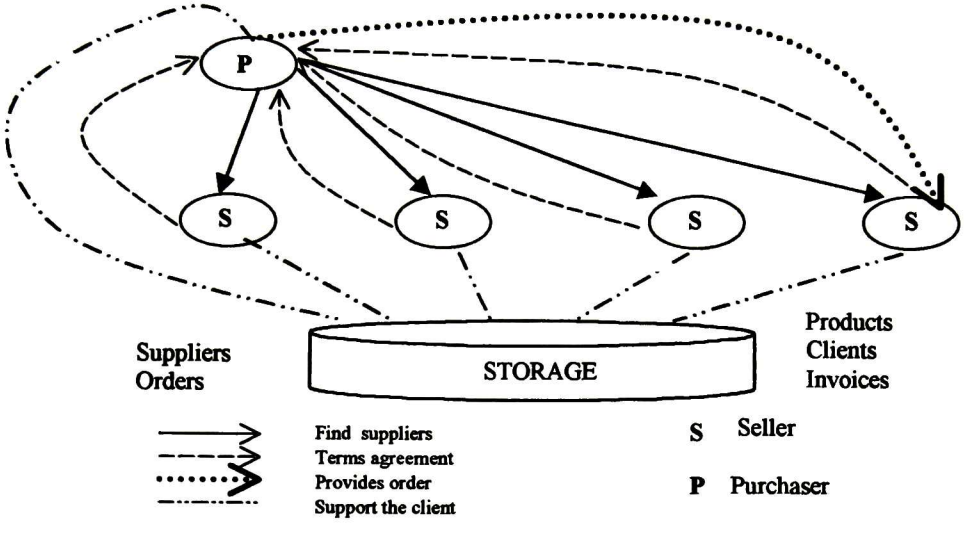


Fig. 5.6 Funcionalidad del protocolo CB

En el proceso de Acuerda términos, los posibles vendedores hacen sus ofertas al comprador, el comprador examina las oferta y se decide por la que más le convenga. En el proceso de soporte al cliente, el vendedor registra la operación con los atributos necesarios, con el fin de tomarlas como referencia en operaciones futuras. La funcionalidad del protocolo CB se muestra en forma gráfica en la Fig.5.5.

5.3.2.1 Diagrama de estados para el protocolo CB

En la Fig. 5.7 se muestra una gráfica que exhibe los estados y las transiciones que representan la funcionalidad del protocolo CB. Esta gráfica constituye el diálogo que sostienen dos clases de agentes: Un agente cliente (Comprador), denotado con la letra A y tres agentes proveedores denotados por B, C, D (vendedores), A_i, B_i, C_i son estados subsecuentes de A, B y C respectivamente; con el fin de encontrar al mejor proveedor y de esa forma adquirir un producto.

En la Fig. 5.7 se identifican las cuatro fases descritas previamente en la Fig. 5.6 las fases son identificadas de la siguiente manera: primero; Encuentra proveedores, paso 1-9 en la Fig. 5.7. segundo; acuerda términos, pasos 10-11 de la Fig. 5.7. tercero; proveer orden, paso 12 en la Fig. 5.7. cuarto; soporte al cliente, pasos 13-14 de la Fig. 5.7.

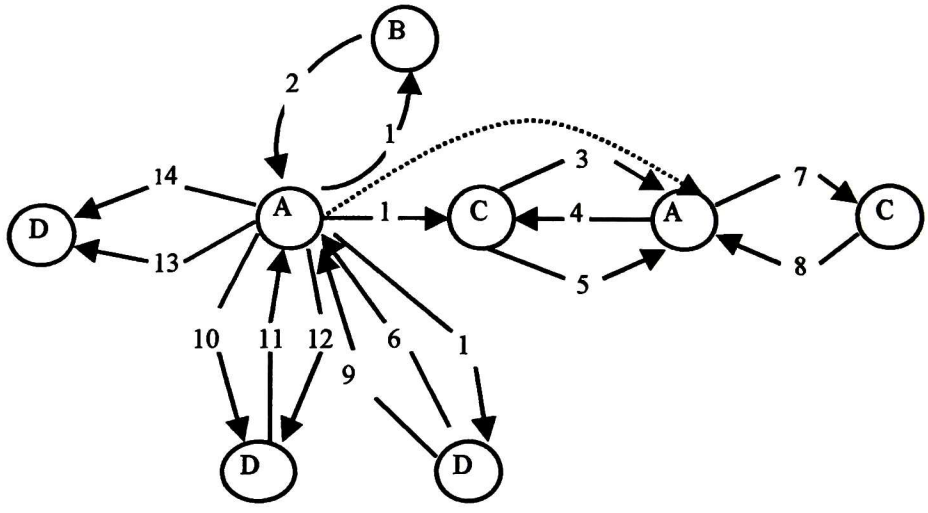


Fig. 5.7 Diagrama de transiciones del protocolo CB

La secuencia de acciones derivadas de la Fig. 5.7 es descrita en la tabla 5.1. Esta tabla muestra las acciones ejecutadas, los agentes participantes y una breve descripción de las acciones correspondientes. Para hacer la descripción presentada en la tabla 5.1 se utilizó el lenguaje LCIASA.

Sequence	Actions
1	Send (A, B, Request(α) $\wedge$ (Bel B friend(A))
	Send (A, C, Request(α) $\wedge$ (Bel C friend(A))
	Send (A, D, Request(α) $\wedge$ (Bel d friend(A))
2	Send (B, A, Abort(α))
3	Send (C, A, Request(tell(x))
4	Send (A, C, Request(replay(x))
5	Send (C, A, Inform(ϕ)) $\wedge$ (Bel A, ϕ)
6	Send (D, A, Inform(ϕ))
7	Send (A, C, Request(subscribe(x))
8	Send (C, A, Abort(α)) $\wedge$ (Bel A, ϕ)
9	Send (D, A, Request(tell(x))
10	Send (A, D, Request(replay(x))
11	Send (D, A, Request(tell(x)) $\wedge$ (Bel a friend(D))
12	Send (D, A Request(tell(x)) $\wedge$ (Bel a friend(D))
13	Send (D, A, Request(insert(x)) $\wedge$ (Bel a friend(D))
14	Send (D, A, Request(insert(x)) $\wedge$ (Bel a friend(D))

Tabla 5.1 Descripción de las acciones para el protocolo CB

En la descripción funcional del protocolo se están describiendo las acciones que los agentes llevan a cabo, para cumplir con el propósito de cada una de las fases mencionadas anteriormente, sin embargo no se hace una descripción explícita del como se realizan éstas acciones. El proceso de intercambiar mensajes para solicitar y ofrecer servicios, incluyendo el proceso de mediación entre los agentes se han implementado considerando algunos aspectos de los trabajos realizados por K. Sycara [48], Tim Finn et al. [50] y Z. Aguirre[1].

5.4 Implementación del protocolo CB

Como se menciona anteriormente el protocolo CB fue creado, agrupando algunos conceptos de mercadeo electrónico, estructuras de coordinación, estándares de comercio electrónico, protocolos de negociación y cooperación; con el propósito de obtener una solución apropiada al problema de comercio electrónico. Es decir, éste protocolo representa la solución que exhibe como evoluciona el sistema en el tiempo.

La implementación es considerada como la traslación del protocolo de interacción que representa la solución del problema que se está resolviendo, a una especificación hecha en algún lenguaje de programación apropiado a ésta aplicación [28].

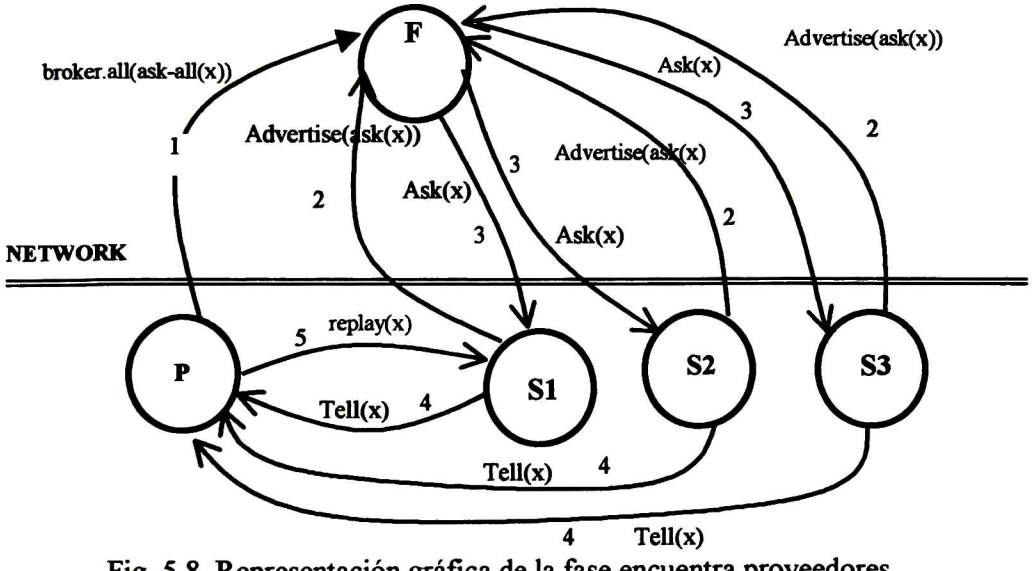


Fig. 5.8 Representación gráfica de la fase encuentra proveedores

En este trabajo de tesis se propone el protocolo CB como la solución al problema propuesto y se propone el lenguaje LCIASA como el lenguaje de comunicación entre agentes para trasladar las acciones descritas en el protocolo a una forma computacional que describa la solución a este problema.
Para ejemplificar la implementación del protocolo CB, se toma la fase del protocolo: encuentra proveedores, la cual se representa en forma gráfica en la Fig. 5.8. En ésta gráfica se hace una descripción, de las principales acciones correspondientes a esta fase. En ésta descripción utiliza algunas de las sentencias del lenguaje LCIASA.

En la gráfica 5.8, el agente comprador (P) anuncia que producto quiere comprar, a través del agente mediador (F), como se muestra en el paso 1 de la Fig. 5.8; el agente vendedor con posibilidades de ofertar el producto demandado por el comprador, ofrece su producto, paso 2 de la figura 5.8; los vendedores proponen al comprador sus requisitos de venta, pasos 3,4; y finalmente el agente comprador selecciona al vendedor que le ofrezca las mejores condiciones de compra, paso 5 de la Fig. 5.8.

Como se mencionó anteriormente en la sección 5.2.2, en la funcionalidad del protocolo CB de identificaron las siguientes 4 fases: Encuentra proveedores, Acuerda términos, proveer orden y soporte al cliente. A continuación se especifica la fase Encuentra proveedores, utilizando, primero: una representación gráfica ver Fig. 5.8 y después se utiliza LCIASA para llevar a cabo la codificación de la secuencia de acciones asociadas a ésta fase del protocolo CB, como se muestra en la Fig. 5.9.

```
Plan Find-suppliers
  Begin
    Variables  Catalogue, P, S1, S2, S3, F : Agents
    Preconditions: KQML protocol
     Begin–Plan
       While (( Catalogue = True) ∧ ¬Eos )
          Send (P,F,  recruit-all (ask-all(x))) ∧ (Bel F Authenticates-Originator (P))
          Do (P, recruit(ask-all(x)))
          Send (S1,F, Advertise (ask (x))) ∧ (Bel S1 Authenticates-Originator (S1))
          Do (S1, Advertise(ask (x))
          Send (S2,F, (Advertise (ask (x))) ∧ (Bel S1 Authenticates-Originator (S2))
          Do (S2, Advertise(ask (x)))
          Send (S3,F, Advertise (ask (x))) ∧ (Bel S1 Authenticates-Originator (S3))
          Do (S3, Advertise(ask (x))
          Send (F,S,  broadcast(ask (x))) ∧ (Bel S2 Authenticates-Originator (F))
          Do (F, ask(x))
          Send (S1,P, Tell(x)) ∧ (Bel P Authenticates-Originator (S1))
          Do (S1, Tell(P))
          Send (S2,P, Tell(x)) ∧ (Bel P Authenticates-Originator (S2))
          Do (S2, Tell(P))
          Send (S3,P, Tell(x)) ∧ (Bel P Authenticates-Originator (S3))
          Do (S3, Tell(P))
          If (Val(S1) >((Val(S2) and Val(S2))) then
          Send (P, S1,  Tell(x)) ∧ ((Bel S1 Authenticates-Originator (P))
          Do (P, Tell(x)
     Do (Cheks-Plan)
   End –Plan
 End
```

Fig. 5.9 Codificación de la fase Encuentra proveedores

En la Fig. 5.9 se muestra el código expresado por medio del lenguaje LCIASA, para la fase encuentra proveedores de la aplicación de comercio electrónico. La codificación expresada en la Fig. 5.9 es considerada como la Implementación de la aplicación propuesta. En la codificación presentada en la Fig. 5.9 se agregan las acciones con las cuales sea posible lograr las propiedades de seguridad y vivacidad y de esa manera asegurar la consistencia de la secuencia de acciones correspondientes a la aplicación.

5.5 Plataforma de Agentes

La plataforma de agentes es un esquema de trabajo en cual permite crear, interpretar, ejecutar, transferir, organizar y finalizar a los agentes, con el propósito de cumplir con una tarea designada.

La traslación a una representación ejecutable de la codificación obtenida previamente, fue implementada utilizando la plataforma de agentes creada por Z.S. Aguirre P.[1] en su tesis de maestría. Este trabajo se desarrolló en forma conjunta con el presente trabajo de tesis, con el propósito de ser utilizado como soporte al mismo, bajo la dirección del Dr. Félix F. Ramos Corchado.

Conclusión

En éste capítulo se hace una propuesta original e innovadora para definir el modelo de interacción entre los agentes, éste modelo representado por el protocolo CB representa la solución dinámica para la aplicación que se está resolviendo. La propuesta consiste en: dada una aplicación a resolver; seleccionar del conjunto de los posibles protocolos de coordinación, cooperación y negociación los mas apropiados para construir un modelo que representa la solución dinámica para esta aplicación específica.
La propuesta que se exhibe en este capítulo, requiere de tener un conocimiento de los posibles protocolos de interacción y de contar con un buen criterio para que la selección de estos protocolos sea apropiada a la aplicación que se está resolviendo. Para obtener un obtener un modelo de interacción independientemente de las personas que deban seleccionar los protocolos adecuados; se propone utilizar los fundamentos de la minería de datos para hacer una clasificación y después una selección de los protocolos de interacción que resulten los mas apropiados al problema que se está resolviendo; para posteriormente construir el modelo que represente la solución dinámica para este problema.

Capítulo 6

Fase de Verificación

Resumen

En este capítulo se presentan los modelos y procedimientos básicos para verificar los Sistemas MultiAgente durante las diversas etapas de su desarrollo, el propósito principal de estos modelos y procedimientos es el de valorar y mejorar la calidad de los Sistemas MultiAgente implementados. Para los fines que se persiguen en ésta tesis, se abordan dos procesos básicos de verificación: Verificación del ciclo de vida y verificación formal, haciendo énfasis en la verificación del ciclo de vida y sugiriendo herramientas para conseguir la verificación formal.

6.1 Introducción

La verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica. De la misma forma, la verificación es el proceso de mostrar que un sistema implementado es correcto con respecto a su especificación. Es decir la verificación implica la valoración de los productos de trabajo para determinar su apego a las especificaciones. En la metodología propuesta se consideran dos tipos de verificación: Verificación de ciclo de vida y verificación formal

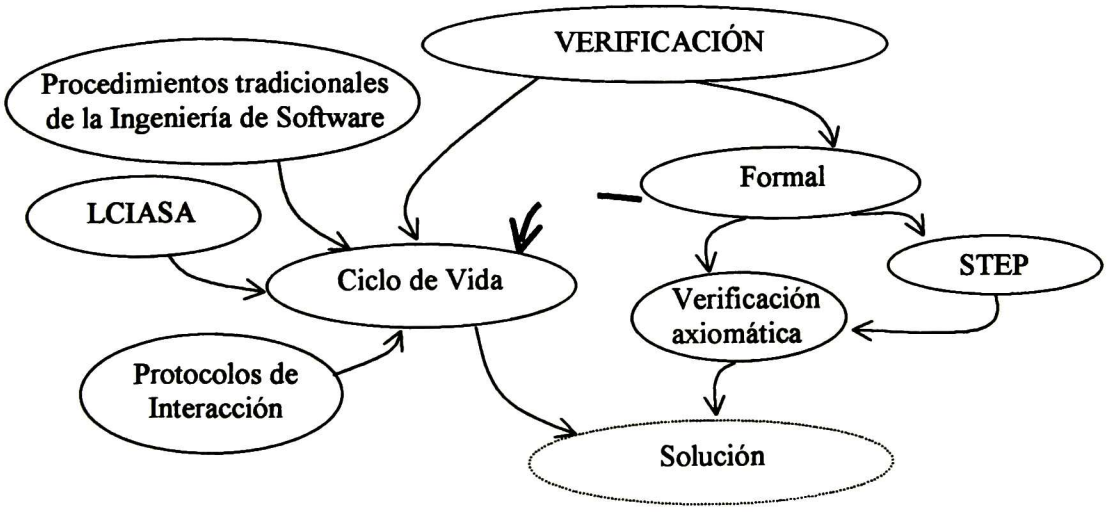


Fig. 6.1 Fase de verificación

La Fig. 6.1 muestra los procesos involucrados para llevar a cabo las actividades propuestas en éste trabajo de tesis para realizar la fase de verificación. Como se mencionó anteriormente, en este trabajo de tesis se da énfasis a la verificación de ciclo de vida. Las acciones principales

que se proponen para realizar la verificación de ciclo de vida son: Imponer mecanismos de seguridad, autenticación y autorización en las acciones que constituyen al lenguaje LCIASA; también se imponen restricciones en los protocolos de interacción con el fin de asegurar la exactitud de las acciones ejecutadas.

6.2 Verificación de ciclo de vida

La verificación de ciclo de vida, es el proceso para determinar el grado en que los productos de trabajo de una fase dada de desarrollo, cumplen con las especificaciones establecidas durante las fases previas. La Fig. 6.2 ilustra las actividades para la verificación del ciclo de vida.

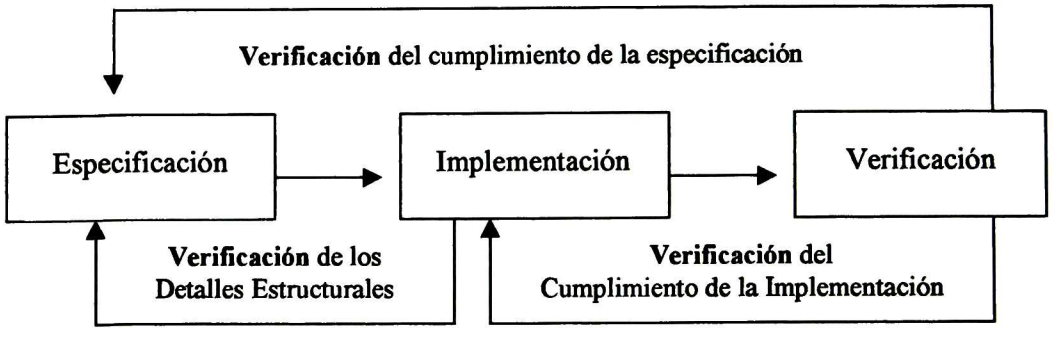


Fig. 6.2 Verificación de ciclo de vida en la metodología

Las principales actividades consideradas para la verificación del ciclo de vida consisten en asegurar las propiedades de vivacidad y seguridad en el sistema MultiAgente construido, las labores emprendidas para afirmar estas propiedades son:

- Definir un conjunto de instrucciones del lenguaje con las cuales se corrobore que las acciones ejecutadas por el sistema sean las esperadas. Como se describió previamente en las acciones de autorización, autenticación y seguridad del lenguaje LCIASA, como se mostró previamente en la tabla 4.8.
- Imponer restricciones en los protocolos de interacción para corroborar que las acciones realizadas sean las esperadas, antes de continuar con la ejecución de las siguientes acciones. Como se mostró previamente en la Fig. 4.15.
- Asegurar que la secuencia de acciones correspondientes a una especificación acordada, posean una estructura y una conclusión adecuada. Como se garantiza en la representación semántica de LCIASA. Lo cual se muestra posteriormente en la fig. 6.2.
- Aplicar las técnicas de verificación clásicas de la Ingeniería de Software [36].

6.3 Verificación formal

La verificación formal es una rigurosa demostración matemática de la concordancia del código fuente con sus especificaciones iniciales.

Una de las pruebas de verificación consiste aplicar arboles semánticos modales para mostrar la satisfaciblilidad de los modelos que representan a los procesos del sistema. Sin embargo este método puede conducir a una explosión de estados, ocasionando que el problema se transforma en un problema de complejidad NP.

Las aproximaciones más exitosas para la verificación de sistemas se pueden dividir en dos clases amplias: axiomática y semántica [14], [8].

6.3.1 Aproximación axiomática

La *aproximación axiomática* requiere de que podamos tomar nuestro programa concreto y de este programa derivar sistemáticamente una teoría lógica que represente el comportamiento del programa. A esta teoría la llamamos la teoría del programa. Si la teoría del programa está escrita en el mismo lenguaje que el de la especificación original, entonces la verificación se reduce al problema de prueba siguiente: mostrar que la especificación es un teorema, es decir es una consecuencia lógica de la teoría del programa [19].

El desarrollo de una teoría del programa es posible por medio de la axiomatización del lenguaje de programación en el cual el sistema está implementado. Hoare formuló un conjunto de reglas como un riguroso sistema axiomático, importando con esto los métodos de la lógica formal, dentro del contexto del razonamiento por medio de programas.

Hoare adoptó un esquema básico de la forma $\{P\}S\{Q\}$ que nos dice que si la aserción P es válida cuando el programa S es iniciado, entonces la aserción Q será verdadera siempre y cuando este termine. El sistema de Hoare, como cualquier cálculo formal consiste de un conjunto de axiomas, junto con un conjunto de reglas de inferencia para derivar teoremas a partir de los axiomas [19].

Hoare utiliza el axioma de asignación: $\vdash \{P[a/x]\}\ x := a\{P\}$ aquí P[a/x] denota la proposición obtenida a partir de P, reemplazando cada ocurrencia de 'x' en P por 'a' Además se tienen las siguientes reglas de inferencia:

i) Reglas de consecuencia if $\vdash\{P\}S\{Q\}$ and Q implica R, Then $\vdash\{P\}S\{R\}$, if P implica Q, and $\vdash\{Q\}S\{R\}$ then $\vdash\{P\}S\{R\}$.

ii) Regla de composición: if $\vdash\{P\}S_1\{Q\}$ and $\vdash\{Q\}S_2\{R\}$ then {P} S1; s2 {R}.

iii) Regla de iteración: if $\{P\wedge B\}S\{P\}$ then $\vdash\{P\}$ while B do $S\{\neg B\wedge P\}$.

A continuación se presenta una aproximación axiomática, para producir una teoría del programa a partir de un sistema axiomático que utiliza las reglas de inferencia del sistema de

Hoare, para lo cual se utiliza la siguiente notación: a; b; c; .... g |- z que es leída " a partir de las premisas a; b; c; ... g puede aseverarse z"

Con la aproximación propuesta se hace una traslación del meta-lenguaje AML al lenguaje LCIASA, con el fin de obtener un lenguaje de programación con una estructura sintáctica bien definida y con una simple interpretación semántica.

Para los propósitos que se persiguen en éste trabajo de tesis se construye un sistema axiomático que toma como base las reglas semánticas de AML, expuestas previamente en la fig. 4.6.

La intención de este sistema axiomático consiste en definir la sintaxis y la semántica del lenguaje LCIASA. Previamente en la sección 4.3.2 de definió la sintaxis de LCIASA utilizando la notación BNF, como lo muestra la Fig. 4.13. Sin embargo ésta representación de la sintaxis de LCIASA está limitada ya que carece de una interpretación semántica dentro del contexto de los Sistemas MultiAgente y puede resultar ambigua.

Esta es la razón principal para crear un sistema axiomático con el cual proporcionar una interpretación semántica al lenguaje LCIASA dentro del marco contextual de los Sistemas MultiAgente.

En la construcción de la aproximación se emplean las reglas semánticas para el lenguaje LA como premisas del sistema axiomático mencionado anteriormente, definiendo un conjunto de reglas de aserción como un sistema deductivo tal que garantice que las especificaciones e implementaciones, hechas con éste, sean una consecuencia de la aplicación de dichas reglas.

Esto es, con el sistema axiomático es posible obtener un programa que represente la solución del problema que se está resolviendo, como consecuencia de aplicar los axiomas y reglas de aserción definidas para éste sistema axiomático, lo cual proporciona un mecanismo adicional de verificación con el cual garantizar la corrección de una especificación o implementación.

La Fig. 6.3 ilustra algunas de las reglas axiomáticas utilizadas para dar una estructura sintáctica y para hacer una interpretación semántica de LCIASA. Esta representación de LCIASA se utiliza, como lo mencionamos previamente, para dar una representación computacional a la especificación e implementación de los modelos y procedimientos definidos en la solución de un sistema MultiAgente específico.

$< M, u> = T \ \vdash < M, u> \models$ true	True-value
$<M, u> \models (Bel\ i, \phi)$ iff $\phi \in bel(\sigma, I(i), u) \vdash$ **B-a**	Bel-action
$< M, u> \models (Do\ i, \alpha)$ iff $action(\sigma, I(i), u) = I(\alpha) \vdash$ **D-a**	Do-action
$< M, u> \models (Send\ i, j, \phi)$ iff $< I(i), I(j), \phi> \in sent(\sigma, u) \vdash$ **S-a**	Send-action
$< M, u> \models \neg\phi$ iff $< M, u> \not\models \phi \vdash$ **N-a**	Neg-action
$< M, u> \models \phi \vee \psi$ iff $< M, u> \models \phi$ or $< M, u> \models \psi \vdash$ **o-o**	or- operator
$< M, u> \models O\phi$ iff $< M, u+1> \models \phi \vdash$ N-o	Next-ϕ
$< M, u> \models \otimes\phi$ iff $u>0$ and $< M, u-1> \models \phi \vdash$ **P-o**	Prev-ϕ

$< M, u> \models \phi \mu \psi$ iff $\exists k \in N$ s.t. $k \geq u$ and $< M, k> \models \psi$ and
$\forall w \in N$ s.t $u \leq w < k$, $< M, w> \models \phi$ |- **D-uc** Do-until-cycle
$< M, u> \models \phi \omega \psi$ iff $< M, u> \models \phi \mu \psi$ or $< M, u> \models \Box \phi$ |- **D-c** Do-cicle
$< M, u> \models \phi \Rightarrow \psi$ iff $< M, u> \models \neg \phi \vee \psi$ |- **I-a** If-action
|- a+b+c...z+A+B...+Y+Z letter
|- 0+1+2..9 digit
|- ++-+/+* Aritm-op
|- $< + = + > + \neq + \geq + \leq$ Rel-op
|- Point
|- comma
r letter |- r identifier
s letter; t identifier |- st identifier
x digit |- x number
x digit |- x number x idenfitier; y Action |- xy Agent-Action
r Agent-Action; m comma; s Bool-Exp |- rms Agent-Belief
j Bool-exp |- if j IF-exp
x Bool- exp |- then x Then-exp
r IF-exp; s Then-exp |- rs If-Then-exp
m identifier |- Plan m heading-of-the-plan
x pre-condition; y declaration-of-variables; w body-of-the plan, z post-conditions |- Begin xywz Block-of the-plan End
x heading-of-the-plan; y block-of-the-plan |- xy Plan

Fig. 6.3 Descripción sintáctica y semántica de LCIASA

El propósito del sistema axiomático creado está limitado a la definición de la sintaxis y semántica de LCIASA y a su utilización para describir los modelos incluidos en las fases de la especificación e implementación de la metodología propuesta. En la tesis no tiene el propósito de aplicar la verificación formal, solo se sugiere el empleo del demostrador automático de teoremas STEP [47].

Conclusión

La verificación es uno de los procesos más importantes en el desarrollo de Sistemas MultiAgente, debido al alto costo que puede llegara a representar la presencia de errores en un sistema MultiAgente. Por esta razón es recomendable adoptar los procedimientos necesarios para poder asegurar la corrección de los sistemas implementados.

Capítulo 7

Conclusiones y Trabajo Futuro

Resumen

En este capítulo se presenta un esbozo de los resultados obtenidos, haciendo hincapié en las aportaciones y trabajos de investigación derivados de éste trabajo de tesis; asimismo se hacen unos breves comentarios acerca del trabajo realizado; también se exhiben las principales conclusiones obtenidas así como algunas sugerencias para continuar con el trabajo en lo futuro.

7.1 Resultados Obtenidos

Como consecuencia de la realización del presente trabajo de tesis se consiguieron diversas aportaciones originales al campo de los sistemas MultiAgente; asimismo se obtuvieron algunas publicaciones en distintos foros de investigación científica. A continuación se hace una breve descripción de los principales resultados obtenidos:

7.1.1 Aportaciones

- Una metodología para el análisis y diseño de Sistemas MultiAgente que plantea un enfoque original e innovador en sus procedimientos y métodos para facilitar la construcción de sistemas de información basados en el paradigma de agente, con la calidad que demanda la industria para este tipo de aplicaciones.
- Un modelo formal para describir a los agentes con el cual es posible modelar y validar sistemas MultiAgente en forma genérica.
- Propuesta de un método de clasificación de los agentes, aunque el método elegido en este trabajo de tesis es simple, se sugiere la utilización de principios de la minería de datos, con los cuales concretar un método preciso para la clasificación de agentes.
- En lenguaje LCIASA, basado en los fundamentos de los sistemas intencionales, lógica temporal y el speech act, para especificar formalmente la interacción entre agentes. La asociación de LCIASA con el Metalenguaje de Agentes AML ofrece una perspectiva innovadora para la implementación de Sistemas MultiAgente. Con LCIASA es posible comprobar la consistencia de una especificación para un SMA, así mismo es posible estudiar el comportamiento, características y capacidades del sistema especificado.
- El protocolo de interacción CB, en donde se propone la selección de los protocolos de interacción más apropiados para crear el modelo que representa una solución apropiada del comportamiento dinámico de la aplicación que se ha de resolver.

7.1.2 Trabajos de investigación

Durante el desarrollo de la tesis se originaron ideas, conceptos, modelos y procedimientos innovadores e interesantes, los cuales sirvieron de soporte para producir un conjunto de trabajos de investigación. Estos trabajos se sometieron en diferentes congresos, simposiums y revistas para su presentación y publicación. A continuación se hace una breve referencia de los trabajos de investigación derivados de esta tesis.

- Se envió el artículo LCIASA: *Lenguaje de "Capacidad de Interacción" de agentes en Sistemas Abiertos* al evento ELECTRO 2000, evento realizado en el Instituto Tecnológico de Chiuhauhua, aceptado y publicado en las memorias del evento, ISSN 1405-172, pp. 337-344.
- Se sometió el artículo *Agents "Interaction Capacity" language,* al evento IASTED International Conference on Applied informatics (AI2001), en Innsbruck, Austria, fue aceptado para su exposición.
- Se sometió el artículo *Agents Interaction Model in Open Systems* al evento 12ª Reunión de Otoño de comunicaciones, computación y electrónica y exposición Industrial IEEE ROC&C'2001, en Acapulco Gro. Mex. Fue aceptado y publicado en las memorias del evento.
- Se sometió el proyecto de investigación *LTLAS: A Language based on Temporal Logic for Agent Systems Specification* al evento Laptec 2002, III congresso de lógica aplicada a tecnología, evento a realizarse del 11 al 13 de Noviembre de 2002 en Sao Paulo, Brasil, el artículo fue aceptado para su presentación en el evento y su publicación en la editorial IOS press.
- Se sometió el artículo *MultiAgent System Methodology Based on a Study of Agents's Interaction* al evento 2ª JIISIC, evento a realizarse del 30 de Octubre al 1 de Noviembre de 2002 en Salvador Bahia, Brasil, artículo aceptado para su presentación y publicación en el evento.
- Se sometió el artículo LCIASA: *A Useful Language for Specification and Verification of Agent-Based Systems*, al evento OPODIS'02, Reims France, 2002.
- Se sometió el artículo *Methodology for MultiAgent Systems Analysis and Design*, al evento 18th ACM Symposium on Applied Computing (SAC 2003), Melbourne Florida, USA, 2003.
- Se sometió el artículo *Formalization of Contract Business Protocol like Mechanism of Coordination to e-Commerce Applications*, al evento ISADS 2002, Guadalajara, Jal. México, 2002, el artículo fue aceptado para su presentación y publicación.

Con los resultados obtenidos se cuenta con un marco de trabajo que facilita el estudio de las aplicaciones actuales y futuras de los sistemas MultiAgente, como lo son: aplicación de realidad virtual, comercio electrónico y en general sistemas de información distribuida.

7.2 Observaciones

El estudio de los sistemas MultiAgente, es un campo de gran importancia en todas las áreas del conocimiento, pero lo es en especial en la Ingeniería de Software, al cual se debe dedicar una gran cantidad de esfuerzos para obtener una metodología orientada Agente, estandarizada, con la cual hacer frente a las aplicaciones actuales y futuras.

La razón principal que remarca la importancia del paradigma de Agente esta dada por las limitaciones que presentan los paradigmas utilizados actualmente en la elaboración de los sistemas de información. Esto es, son bien conocidos los límites del paradigma cliente - servidor utilizado por el paradigma Orientado a Objeto para manejar las soluciones en forma coordinada, cooperativa y negociada que requieren las aplicaciones desarrolladas en contextos distribuidos abiertos.

Basado en estudios recientes en esta área de la Ingeniería de Software, queda expuesto que el paradigma Orientado Agente se presenta como una opción prometedora para obtener soluciones apropiadas a los problemas actuales y futuros. Por ésta y otras razones es necesario dedicar grandes esfuerzos enfocados al estudio del paradigma Orientado Agente.

7.3 Conclusiones

En este trabajo se propone una metodología formal para Análisis y diseño de Sistemas MultiAgente, en la metodología se plantea la integración de los conceptos de mayor importancia asociados al paradigma de agentes, organizándolos de una manera ordenada y coherente para obtener una regulación que facilite la construcción de Sistemas MultiAgente.

La metodología propuesta esta enfocada hacia un adecuado desarrollo de aplicaciones complejas, basadas en el paradigma de agente. Como se describió en las secciones previas, la metodología permite una especificación de los sistemas MultiAgente más expresiva, legible, veraz y fácil de utilizar y entender. Esta metodología facilita el análisis y diseño de sistemas a través de la aplicación de la bien definida secuencia de pasos que se proponen en esta.

Esta metodología es apropiada para desarrollar, entre otras, las siguientes aplicaciones cooperativas: operaciones de compra - venta de productos, transacciones bancarias, comercio electrónico, aplicaciones industriales, Aplicaciones educativas, etc.

La metodología propuesta ofrece las siguientes ventajas: primero, la metodología tiene asociado un modelo formal, el cual, hace posible especificar y validar las acciones, propiedades y características un sistema. Segundo, la metodología emplea los las técnicas de modelado y los protocolos de interacción que son aceptados como estándares en el ámbito de los sistemas MultiAgente. Tercero, la utilización de CORBA como soporte básico para la comunicación, facilita la herencia, interoperabilidad y la implantación de nuevas aplicaciones.

Esta metodología tiene su principal aplicación en los Sistemas Multiagente que son tratados como sistemas intencionales, es decir, sistemas en donde a los agentes se les atribuyen estados mentales como creencias e Intenciones. Sin embargo dada la generalidad de esta

metodología, es posible aplicarla a la solución de la mayoría de los sistemas basados en el paradigma de agente.

Finalmente se concluye que la metodología para el análisis y Diseño de Sistemas MultiAgente presentada en este trabajo de tesis ofrece un enfoque original e innovador en su organización, conceptos, procedimientos y aplicación con el fin de proporcionar una norma para el estudio y desarrollo simple y amigable de aplicaciones basadas en los fundamentos del paradigma de agente; asimismo, ésta metodología promete hoy en día y en lo futuro, muchas expectativas para alcanzar una adecuación apropiada hacia los nuevos modos del procesamiento de la información.

7.4 Trabajo Futuro

La metodología propuesta ha sido expuesta en su mayor parte, sin embargo no se ha concretado en su totalidad, quedando como trabajo futuro los siguientes problemas:

- Desarrollar una notación específica para la metodología propuesta.
- Aplicar técnicas de minería de datos para lograr una clasificación concreta de los agentes.
- Hacer un Análisis comparativo de las diferentes metodologías para el Análisis y diseño.
- Encontrar de manera automática los modelos teóricos y los protocolos de interacción más apropiados a una aplicación específica.
- Aplicar herramientas para la verificación formal de los modelos derivados de las fases de la especificación e implementación.
- Formalizar algunos de los modelos conceptuales (generadores de código, traductores, etc.).
- Definir las métricas para cuantificar el paradigma.

Bibliografía

[1] Aguirre P. Z.S., Modelado dinámico de organizaciones para el trabajo cooperativo, M. Sc. Thesis degree, CINVESTAV-IPN Guadalajara Unit, México 2001.

[2] Barrenechea, Mark J., e-BUSINESS OR OUT OF BUSINESS, Mc. Graw-Hill, 2001.

[3] Barthes, J.P. MultiAgent Systems, International Symposimun on Advanced Distributed Systems, Guadalajara, Jal. March, 2000.

[4] Bigus J., Constructing Intelligent Agents with Java™ A programmer's Guide to smarter applications, John Wiley & Sons, Inc. 1998.

[5] Booch, Object-Oriented Analysis and Design with applications, Addison wesley, 1996.

[6] Clark E., Grumberg O., Peled D., Model Checking, The MIT Press Cambridge Massachusetts London England, 1999.

[7] Donovan J., system programming, McGraw-Hill International editions, 1972.

[8] Farías M.N., LCIASA: Lenguaje de "Capacidad de Interacción" de Agentes en Sistemas Abiertos. M. Sc. Thesis degree CINVESTAV-IPN Guadalajara Unit, México 2000.

[9] Farías M.N., Ramos C.F.F., LCIASA: Lenguaje de "Capacidad de Interacción" de Agentes en Sistemas Abiertos: LCIASA, *ELECTRO 2000* del Instituto Tecnológico de Chihuahua. Publicado en las memorias del evento, pp. 337-343. , México, 2000.

[10] Farías M.N. and Ramos C.F.F. Agents Interaction Model in Open Systems, IEEE ROC&C'2001, 12 Reunión de Otoño de Comunicaciones, Computación y Electrónica, Acapulco Gro. México 2001.

[11] Farías M.N., Ramos C.F.F., MultiAgents Systems Methodology Based on a Study of Agent's Interaction. Proc. of the 2ª Jornada Ibero Americana de Engenharia de Software e Engenharia de Conhecimento. Salvador Bahia Brasil, 2002.

[12] Farías M.N. LTLAS: a Language Based on Temporal Logic for Agents Systems Specification, Third congress of Logic Applied to Tecnologhy, IOS Press Holland. Sao Paulo, Brasil, 2002.

[13] Farías M.N. Ramos C.F.F, Aguirre Z., Larios R.V.M., Formalization of the Contract Business Protocol like Mechanism of Coordination to e-Commerce Applications, Proc. of the Second IEEE International Symposium on Advanced Distibuted Computing Systems: ISADS 2002, Guadalajara, Jal. México 2002.

[14] Farías M.N. Ramos C.F.F. Larios R.V.M., LCIASA: A Language for Specification and Validation of Agent-Based Systems Interaction. Proc. of the 6th International Conference On Principles Of Distributed Systems OPODI'S 02, Reims,France, 2002.

[15] Farías M.N., Ramos C.F.F. Methodology for MultiAgent Systems Analisis and Design, sometido al 18th ACM Symposium on Applied Computing (SAC2003), Melbourne Florida, USA. 2003.

[16] Ferber Jacques, Multi-Agent Systems: An introduction to Distributed Artificial Intelligence, Addison-Wesley, 1999.

[17] Foundation for intelligent Physical Agents. Specifications. 1997. Available from http://www.fipa.org.

[18] Galier Jean H., *Logic for Computer Science: Foundations of Automatic Theorem Proving*, John Wiley & Sons, 1987.

[19] Galton A. Temporal Logics and their applications, Academic Press, 1987.

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

[21] G.M.P. O' Hare, and N.R. Jennings, *Foundations of distributed Intelligence*, Jhon Wiley & Sons Inc.

[22] Huhns M.N. & Singh M.P., *Readings in Agents*, Morgan Kaufmann publishers, Inc. 1998.

[23] Gonzalez Torres R.E., " *Lógica temporal: Apuntes del curso de lógica*" CINVESTAV IPN, Unidad Guadalajara Jal. México, 2002.

[24] Haddadi A., Sundermeyer K. " Chapter 5 –Belif-Desire-Intentions Agent Architectures , " *Foundations Of Distributed Artificial Intelligence*., G.M.P. O'Hare and N.R. Jennigs (eds) Jhon Wiley & SonsInc., pp 169-186 1996.

[25] Jackson M.A. *System Development*, Prentice Hall, 1983.

[26] Jacobsen Ivar, Object oriented development in an industrial environment, OOPSLA'87, publicado como ACM SIGPLAN 22, (Dic. de 1987), pp. 183-191.

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

[28] Jennings N.R. and Wooldridge M., Agent Oriented Software Engineering, proceedings of the 9th European Workshop on modelling Autonomous agents in Multi-Agent world: Multi-Agent Software Engineering (MAMAV-99).

[29] Kinny D. And Georgeff M., Modelling and Design Multi-Agent Systems, Intelligents Agents III, Muller, Wooldridge, Jennings (Eds.), Springer pp. 1-20, 1996.

[30] Koth, Silverchatz, *Database System Concepts*, Mc Graw-Hill, pp. 68-81 1991.

[31] Maes P. (ed), Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back. London, the MIT Press.

[32] Maldonado T.A. , TUTORIAL: Comercio Electrónico Interempresarial, ISADS 2000, Guadalajara, Jal. México 2000.

[33] Malone W.T., Modelling coordination in organization and markets, Manage sci, 33 1317-1332.

[34] Manna Z., Pnueli A., Temporal Verification of Reactive Systems: Safety, Springer, 1995.

[35] Manna Z.& Waldinger R., The logical Basis for coputer programming, Addison Wesley, 1985.

[36] Marciniak J.J., Encyclopedia of Software Engineering, Jhon Wiley & sons, Inc. 1994.

[37] Michael Wooldidge, Jorg P. N.R. Jennings, *Intelligent Agents*, Vol I, Springer Verlang. 1994.

[38] Open Buying on the Internet (OBI)™, Technical specification, release v2.0, from the OBI Consortium, 1999.

[39] Pope Alan, The CORBA Reference Guide: Understanding the Common Object Request Broker Architecture, Addison Wesley 1998.

[40] Pressman R. Software Engineering; a practitioner's approach, Mc. Graw-Hill, 1997.

[41] Rao A.S and Georgeff, " A model-theoretical approach to the verification of situated reasoning systems" Proc. Of the 13th Int. Joint on Artificial Intelligence, 318-324 Chambery, France.

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

[43] Ross D. "Applications and extentions of SADT", IEEE computer, vol 18, no. 4, 1984, pp. 25-35.

[44] Rumbaugh J. et al, The unified modeling language. Reference manual, addison Wesley, 1999.

[45] Rumbaugh J., et al, Object-Oriented Modelling and Design, Prentice Hall, Inc. 1991.

[46] Seen, Analysis and design of information systems, Mc Graw Hill, 1990.

[47] STEP Automatic theorem demonstrator developed by the Stanford University USA, available from http://rodin.stanford.edu

[48] Sycara, K., "Brokering and Matchmaking for coordination of Agent Societies: A Survey" In Coordination of Internet Agents, A. Omicini et al.(eds.), Springer., 2001.

[49] The International Engineering Consortium, IBM Electronic Commerce, Thomas Group, 2001, Available from http://www.iec.org.

[50] Tim Finn, Don Mc Kay, Rich Fritzon, AN OVERVIEW OF KQML: A_KNOWLWDGE QUERY AND MANIPULATION LANGUAJE, University of Maryland, Baltimore MD 21228.

[51] Ward P.T.y S.J. Mellor, Structured development for real time systems, tres volumenes, Yourdon Press, 1985.

[52] Warnier J.D., Logical construction of programs, Van Nostrand Reinhold. 1981.

[53] Weiss G. , Multiagent Systems A Modern Aproach to Distributed Artificial Intelligence, The MIT Press Cambridge, Massachusetts London England, pp. 79-120, 2000.

[54] Witen I., Frank E., Data Mining: Practical Machine learning tools and techniques with JAVA implementations, Academic Press, 2000.

[55] Wooldridge,M., "Chapter 10- Temporal Belief logic for modeling Distributed Artificial Intelligence systems," Foundations od Distibuted Artificial Intelligence, G.M.P. O'Hare and N.R. Jennings (esd), Jhon Wiley & Son.

[56] Yourdon Inc, Yourdin System Method, Yourdon Press computing series, 1993.

Apéndice A Glosario

- ABSTRACCIÓN Proceso para definir un modelo simplificado del sistema, el cual enfatiza algunos detalles o propiedades mientras que ignora otros.
- ACL Lenguaje de comunicación entre agentes
- ACTOR Concepto pionero de programación concurrente
- AGENTE En su uso mas general se conceptúa como un sistema de cómputo autónomo, reactivo, pro-activo y auto–contenido. También es considerado en términos de conceptos más humanos como las creencias, deseos e intenciones.
- AGENTE CREIBLE Es un agente que generalmente es representado en un juego de computadora o en alguna clase de ambiente virtual.
- AGENTE DE INTERFACE es un programa de computadora que emplea técnicas de Inteligencia Artificial para proporcionar asistencia a un usuario que esta tratando con alguna aplicación.
- AGENTE SITUADO Agente que percibe y actúa en el medio ambiente en el que está inmerso.
- AGENTE DE SOFTWARE No es un agentes implementado por software. Más bien es un Agente que opera como sensor y actúa en un ambiente de software tal como UNIX.
- ARQUITECTURA Metodología particular para la construcción de agentes, típicamente incluye definiciones de software, estructuras de datos y operaciones sobre las estructuras.
- ARQUITECTURA BDI Es una arquitectura que contiene una representación explícita de creencias, deseos e intenciones. Las creencias son consideradas como la información que el agente tiene de su medio ambiente, las cuales pueden ser falsas. Los deseos son todas aquellas cosas que el agente quisiera realizar, los deseos no son todos consistentes, y no esperamos que el agente actúe sobre todos ellos. Las intenciones son aquellas cosas a las que el agente se compromete a realizar.
- ARQUITECTURA DELIBERATIVA Arquitectura que se basa en una representación interna dada por un modelo simbólico y una manipulación simbólica.
- ARQUITECTURA HIBRIDA Arquitectura que trata de cazar las técnicas de la Inteligencia Artificial simbólica con otras aproximaciones para agentes.
- ARQUITECTURA REACTIVA Arquitectura que no emplea clase alguna de modelo simbólico central del mundo.
- AGENTE CREIBLE Es un agente que generalmente es representado en un juego de computadora o en alguna clase de ambiente virtual.

- ARQUITECTURA DE PIZARRON Es una clase de arquitectura en la cual una colección de fuentes de conocimiento se comunican escribiendo en una estructura de datos accesible en forma global, conocida como pizarrón.
- AUTONOMIA El hecho de asumir que el agente nunca actúa sin la intervención humana directa u otra intervención y que tienen alguna clase de control sobre su estado interno.
- CONATIVO Hacer con deseo
- COORDINACION Proceso para asegurar que las partes de un problema sean incluidas en al menos un agente, para completar la solución de un problema.
- ESTADO COGNITIVO Estado interno de un sistema Intencional, la colección de creencias, deseos e intenciones, que caracterizan a un agente en un instante dado.
- ESTADO MENTAL estado Cognitivo.
- EPISTEMICO Hacer con conocimiento
- FACILITADOR Interfaz Común para los agentes
- FRAMEWORK Conjunto de principios modelos y procedimientos que conforman una estructura básica para proporcionar el soporte con el cual realizar una actividad específica afín a los fundamentos que definen esta estructura.
- INTERACCIÓN Diálogo que sostienen los agente por medio de un intercambio de mensajes estructurados, para lograr un propósito o meta establecida.
- LENGUAJE DE AGENTE Lenguaje de programación que considera la noción de agente, ejemplo telescript, Agente 0.
- LOGICA MODAL Lógica de necesidad y posibilidad, las técnicas de la lógica modal se han usado para formalizar nociones mentales así como aspectos temporales.
- LÓGICA TEMPORAL Formalismo para describir secuencias de transiciones entre estados, en un sistema reactivo. Es decir, define el comportamiento del sistema a través del tiempo.
- METODOLOGÍA Ciencia que estudia los medios o procedimientos para indagar o exponer verdades científicas.
- MODEL CHECKING Técnica para la verificación de sistemas concurrentes de estado finito.
- NEGOCIACION Es el proceso para alcanzar un estado que es mutuamente conveniente a un conjunto de agentes, esta íntimamente relacionado con la coordinación.
- ONTOLOGIA Entidad computacional, un recurso conteniendo conocimiento acerca de que "conceptos" existen en el mundo y como estos se relacionan con otros
- PLAN es un representación de un curso de acciones que cuando son ejecutadas conducen a alcanzar un objetivo, los planes pueden involucrar a varios agentes.
- PRO –ACTIVO Capaz de tomar la iniciativa, no solamente manejado por eventos, capaz de generar metas y actuar racionalmente para lograrlas.

- PROCESO DE SOFTWARE Marco de trabajo que engloba las tareas que se requieren para construir Software de alta calidad.
- PROGRAMACIÓN ORIENTADA-AGENTES Una aproximación para la construcción de agentes la cual propone la programación de agentes en términos de nociones mentales, como deseo, creencia e intención.
- PROTOCOLO DE COOPERACION Es un protocolo que define como un grupo de agentes trabajan juntos para alcanzar un objetivo. El mas conocido es el contract net.
- SISTEMA INTENCIONAL Sistema cuyo comportamiento se puede predecir o explicar atribuyendo al sistema actitudes como creencias, deseos e intenciones, junto con un grado de racionalidad.
- SISTEMA MultiAgente Red débilmente acoplada entidades autónomas que trabajan en conjunto para resolver un problema que está más allá de las capacidades individuales de un agente.
- TEORIA DEL SPEECH-ACT Teoría pragmática de la comunicación. Los axiomas principales de la teoría del speech-act se basan el hecho de que las expresiones comunicativas son acciones ejecutadas por un emisor y con la finalidad de iniciar un cambio en el estado mental de algún receptor.
- VERACIDAD Asume que un agente no reconoce información falsa.
- VERIFICACION DE CICLO DE VIDA Proceso para determinar el grado con el cual los productos de trabajo de una fase determinada del ciclo de desarrollo, cumplen con las especificaciones de las fases previas.
- VERIFICACIÓN DEDUCTIVA Se refiere al uso de axiomas y reglas de prueba para comprobar la validez de los sistemas.
- VERIFICACIÓN FORMAL demostración matemática rigurosa de que el código fuente cumple con sus requisitos.

Apéndice B Notación

A El agente A
Ag Conjunto de agentes
I_A estados de información de los agentes
S_A estados estratégicos o intencionales de A
V_A estado de evaluación para el agente A
$\Diamond$ " Es posible que" o posiblemente
$\Box$ " es necesario que" o necesariamente
$P\Box$ " fue necesario que"
$\forall$ para todo
$\exists$ Existe
$\neg$ Negación (not)
$\wedge$ Conjunción (and)
$\vee$ Disyunción (or)
$\beta^0 \in BS$ conjunto de creencias iniciales del agente
A : $BS \rightarrow Ac$ función de Acción del agente
M : $BS \rightarrow \wp(Mess)$ función de generación de mensajes del agente
η : $BS \times Ac \times \wp(Mess) \rightarrow BS$ función del siguiente estado del Agente.
(Bel *i* ϕ) El agente *i* cree ϕ
(Send *i j* ϕ) El agente *i* envió al agente *j* el mensaje ϕ
(Do i α) El agente i ejecuta la acción α
$O\phi$ El siguiente ϕ
$\otimes\phi$ El último ϕ
$\phi \mu \psi$ ϕ hasta que μ (no estricto)
$\phi \omega \psi$ ϕ a menos que ψ
$\upsilon\phi$ ϕ hasta ahora
$\blacklozenge\phi$ ϕ alguna vez
$\alpha\phi$ ϕ antes
$\phi\beta\psi$ ϕ Cuando ψ
$\phi \xi \psi$ ϕ desde que ψ

Centro de Investigación y de Estudios Avanzados del IPN

Unidad Guadalajara

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: METODOLOGIA PARA EL ANÁLISIS Y DISEÑO DE SISTEMAS MULTIAGENTE del(a) C. Nicandro FARÍAS MENDOZA el día 11 de Noviembre de 2002 .

Dr. Luis Ernesto LÓPEZ MELLADO
Investigador Cinvestav 3A
CINVESTAV GDL
Guadalajara

Dr. Federico SANDOVAL IBARRA
Investigador Cinvestav 2C
CINVESTAV GDL
Guadalajara

Dr. Félix Francisco RAMOS CORCHADO
Investigador Cinvestav 2A
CINVESTAV GDL
Guadalajara

Dr. Jean-Luc KONING
Vice-Director of LCIS
Research Laboratory
Tecnological University of Grenoble
Francia

Dr. Victor Manuel LARIOS ROSILLO
Investigador Titular
Universidad de Guadalajara
Zapopan