



**FACULTAD DE INFORMATICA  
UNIVERSIDAD POLITECNICA DE MADRID**

**Trabajo de Investigación:**

**Elaboración de una Aproximación  
Metodológica para el desarrollo de Software Orientado  
a Sistemas Multiagente**

**Tutor: Doctor Francisco Javier Segovia**

M.Ing. Jorge Salvador Ierache

[jierache@yahoo.com.ar](mailto:jierache@yahoo.com.ar)

JUNIO 2003

## ÍNDICE

1. Capítulo I. Introducción .....	6
1.1 Introducción.. .....	7
1.2 Descripción de la Composición del Trabajo .....	7
2. Capítulo II. Fundamentos Teóricos .....	9
2.1 Sistemas Basados en Agentes .....	12
2.2 MaSE .....	12
2.2.1 Introducción .....	12
2.2.2 Capturar metas .....	13
2.2.3 Transformar metas en roles .....	14
2.2.4 Aplicar casos de uso.....	15
2.2.5 Crear Clases Agente.....	16
2.2.6. Detalles de Diseño .....	16
2.2.7 Construir conversaciones .....	17
2.2.8 Ensamblado de agentes.....	17
2.2.9 Arquitectura de Agente Reactivo .....	18
2.2.10 Arquitectura de Agente basado en Conocimientos .....	19
2.2.11 Arqutectura de Planeamiento.....	20
2.2.12 Arquitectura BDI.....	21
2.2.13 Desarrollo del Sistema.....	22
2.3 Resumen de Metodologías Orietadas a Agentes .....	25
2.3.1 Análisis y Diseño Orientado a Agentes .....	25
2.3.2 Técnica de modelado de agentes para sistemas de agentes BDI .....	25
2.3.3 Método basado en escenarios para sistemas MultiAgentes (MASB) ....	26
2.3.4 Metodología CoMoMAS .....	26
2.3.5 Metodología MAS-CommonKADS .....	26
2.4 Extensiones de UML para el modelado de agentes.....	26
2.4.1 Diagramas de Secuencia.....	26
2.4.2 Línea de Vida de un Objeto .....	27
2.4.3 Activación.....	27
2.4.4 Mensajes y Estímulos .....	28
2.4.5 Aplicación de diagramas de secuencia a Agentes.....	28
2.4.6 Diagramas de Colaboración.....	29
2.4.7 Aplicación de Diagramas de Colaboración a Agentes.....	30
2.4.8 Diagrama de Actividades .....	30
2.4.9 Diagramas de Estados.....	32
3. Capítulo III. Aplicación en robotica " juego de la escondida" .....	34
3.1 Introducción.. .....	34
3.2 Realidad del Juego .....	34
3.3 Modelo del Juego (abstracción de la realidad).....	34
3.4 Modelización . .....	34

3.5 Descripción de los agentes.....	34
3.5.1 Tipo de Agentes .....	34
3.5.2 Percepciones.....	34
3.5.3 Acciones.....	35
3.5.4 Descripción de metas .....	35
3.5.5 Descripción del ambiente.....	35
3.5.6 Temporalidad .....	36
3.5.7 Localidad .....	36
3.5.8 Procesos.....	36
3.5.9 Población.....	36
3.6 Descripción del escenario físico.....	36
3.7 Modelado utilizando la metodología MaSE .....	37
3.7.1 Casos de uso.....	38
3.7.2 Diagramas realizados aplicando Agent Tools .....	40
3.7.2.1 Diagrama jerárquico de metas .....	40
3.7.2.2 Diagramas de secuencia (cazador).....	41
3.7.2.3 Diagramas de secuencia (presa).....	41
3.7.2.4 Diagrama de roles .....	42
3.7.2.5 Diagramas de tareas (contar).....	43
3.7.2.6 Diagramas de tareas (deambular).....	44
3.7.2.7 Diagramas de tareas (encontrar escondite).....	45
3.7.2.8 Diagramas de tareas (volver a la base) .....	46
3.8 Implementación.....	47
4. Capítulo IV Propuesta de integración de ingeniería del Conocimiento en Metodologías multiagentes .....	50
4.1 Modelización del Conocimiento en un Ambiente Multiagente .....	50
4.2 Adquisición del Conocimiento de los Roles de los Agentes .....	51
4.3 Conceptualización de los Agentes.....	52
4.3.1 Análisis de los Conocimientos .....	52
4.3.2 Identificación, Comparación y Categorización de Conceptos .....	53
4.3.3 Identificación de las Relaciones entre Conceptos.....	53
4.3.4 Identificación de los Conocimientos Estratégicos .....	53
4.3.5 Identificación de los Conocimientos Tácticos.....	54
4.3.6 Identificación de los Conocimientos Fácticos.....	54
4.3.7 Síntesis de Conocimientos.....	55
4.3.8 Síntesis – Modelo Estático.....	55
4.3.9 Modelo Dinámico (Modelo de Procesos) .....	55
4.3.10 Mapa de Conocimientos .....	56
4.3.11 Comprobación de la Conceptualización.....	57
4.4 Formalización de Conocimientos de los Agentes.....	57
4.4.1 Selección de Formalismos .....	57
4.4.2 Formalización de los Conocimientos en reglas de producción .....	57

4.4.3 Modelado de la base de conocimiento de los Agentes .....	57
4.4.4 Formalización de los Conocimientos en Marcos .....	58
4.4.5 Formalización de los Conocimientos en Procedimientos .....	59
4.4.6 Guiones .....	59
4.4.7 Sistemas de Producción .....	59
4.4.8 Representación de Conocimientos en el Sistema de Producción.....	59
4.4.9 Estrategia de Control .....	60
4.4.10 Representación de Conocimientos sobre la solución del problema...	60
4.4.11 Inferencia en el sistema de producción .....	60
4.4.12 Formalización del conocimiento estratégico .....	60
4.5 Modelado del Conocimiento .....	61
4.5.1 Capturando y estructurando Objetivos.....	63
4.5.2 Determinando Roles.....	63
4.5.3 Modelado de roles para agentes .....	64
4.5.4 Clases de Agentes .....	65
4.5.5 Ensamblado de Clases de Agentes .....	65
4.5.6 Diseño del Sistema Multiagente .....	65
5. Capítulo V Conclusiones y futuras líneas de investigación .....	67
5.1 Conclusiones del Trabajo .....	67
5.2 Futuras líneas de Investigación.....	67
6. Capítulo VI Bibliografía .....	69

# **Capítulo I**

## **Introducción**

# 1. Capítulo I

*Para comprender la inteligencia hay que comprender cómo se adquiere , se representa y se almacena el conocimiento; cómo se genera y se aprende el comportamiento inteligente; cómo se desarrollan y usan las motivaciones, las emociones y las prioridades, cómo las señales sensoriales son transformadas en símbolos para aplicar la lógica, para razonar sobre el pasado y para planificar el futuro , y cómo los mecanismos de la inteligencia producen los fenómenos de la ilusión, las creencias, las esperanzas, los temores y los sueños, incluso la bondad y el amor .*

*Comprender estas funciones en un nivel fundamental sería un logro científico de la misma escala que la física nuclear, la relatividad, genética molecular (James Albus febrero 1995).*

## 1.1 Introducción

Los actuales sistemas de información sufrirán transformaciones al abandonar su rol pasivo y pasar a ser activos para tomar decisiones. Un agente es normalmente un asesor de la persona real, a quien sustituye, anticipándose a sus requerimientos. El paradigma de agente nos permite modelar actores con conocimientos, creencias, percepciones, metas, intenciones y obligaciones.

El presente trabajo presenta los conceptos generales de sistemas basados en agentes y las metodológicas aplicadas en ambientes multiagente, con especial interés en la metodología MaSE, se implementa el juego de escondida el que se desarrollo empleando tres plataformas de agentes robot montadas sobre el. kit de Lego Mindstorms Robotic Invention Systems, aplicando la metodologia MaSE , para luego proponer la incorporación de nuevas etapas en la metodológica empleada bajo una visión cognitiva del sistema multiagente en una arquitectura basadas en conocimiento, a través de la incorporación de Ingeniería en Conocimiento a la metodología MaSE

Se presenta la implantación de un ambiente multiagente aplicando la Metodología MaSE, para tal fin se implementa el juego de escondida el que se desarrollo empleando tres plataformas de agentes robot montadas sobre el. kit de Lego Mindstorms Robotic Invention Systems. [1]

Se propone la incorporación de nuevas etapas en metodológica empleada para la implementación de arquitectura basadas en conocimiento a fin de considerar los conocimientos estratégicos tácticos y fácticos del sistema multiagente durante su conceptualización continuando con su formalización a través de marcos.

Para el desarrollo de este trabajo se a recurrido a metodología MaSE (Multiagent Systems Engineering) del AFIT (Air Force Institute of Technology) Agent Lab empleada para el desarrollo de aplicaciones basadas en sistemas Multiagente ,con la particularidad de realizar una adaptación a aplicaciones de Multiagente robóticos los que se implementan sobre una plataforma RCX correspondiente al kit de Lego Mindstorms Robotic Invention Systems.

Este equipamiento es utilizado por diversas universidades del mundo, participando en su desarrollo el MIT. Si bien existen otras plataformas para la implantación de robots más costosas y con mayores capacidades, la selección de ésta no sólo obedeció al bajo costo sino a su aplicación en distintas universidades.

Para la aplicación y demostración de un ambiente multiagente agente con distintos roles se selecciona un juego clásico como lo es la escondida. Si bien en mi época se gritaba "piedra libre", hoy mis hijos dicen "pica", no obstante el juego es el mismo y varias generaciones se divirtieron con él, este se implementa básicamente con tres robots, dos que juegan a esconderse y otro que se dedica a buscarlos.

Desde el punto de vista académico el trabajo realizado también incluye la aplicación de metodología, técnicas, herramientas, desarrollo del software de aplicación para cada agente, pruebas y la construcción del escenario es decir la cuadra del barrio donde solíamos jugar a la escondidas, en nuestro caso ocupa sólo una superficie de aproximadamente 6 metros cuadrados con obstáculos y

lugares para esconderse los cuales deben ser reconocidos por nuestro robot participante durante el juego, al igual que el área donde se debe cantar "piedra libre" o "pica".

## **1.2 Descripción de la Composición del Trabajo**

Se presenta en el capítulo dos, los conceptos generales de sistemas basados en agentes y las metodológicas aplicadas en ambientes multiagente, con especial interés en la metodología MaSE.

El capítulo tres se presenta la implantación de un ambiente multiagente aplicando la Metodología MaSE, para tal fin se implementa el juego de escondida el que se desarrollo empleando tres plataformas de agentes robot montadas sobre el kit de Lego Mindstorms Robotic Invention Systems

En el capítulo cuatro se propone la incorporación de nuevas etapas en metodológica empleada para la implementación de arquitectura basadas en conocimiento a fin de considerar los conocimientos estratégicos tácticos y fácticos del sistema multiagente.

Finalmente en el capítulo cinco se presentan las conclusiones y futuras líneas de investigación, y en el capítulo seis se presenta la bibliografía consultada .

# **Capítulo II**

## **Fundamentos Teóricos**

## 2. Capítulo II

### 2.1 Sistemas Basados en Agentes

Las definiciones básicas de agentes presentan el concepto como algo autónomo que percibe el entorno a través de sensores de visión, tacto, temperatura, entre otros y actúan en él a través de efectores (elementos que facilitan su actuación, brazos robot, mecanismos de movilidad, etc.).

Dentro de las clasificaciones más generales podemos decir que básicamente nos encontramos en este campo con agentes de software los cuales se presentan como específicos para tareas de información, entretenimiento o virus, además se presentan dentro de los agentes de computación los dedicados a modelar la vida artificial. Otro campo es el de agente robótico sobre el cual se ocupa en gran parte de este trabajo.

Un sistema basado en agentes, es aquel en el cual la principal abstracción utilizada es un *agente*. Los agentes software son probablemente, el área de la tecnología de la información que crece de manera más rápida, según la propuesta de Wooldridge y Jennings [2] Existen dos nociones de agente: una débil y otra fuerte.

- **Definición débil:** un sistema computacional hardware o software que goza de las siguientes propiedades:

- **Autonomía:** los agentes operan sin una directa intervención de humanos u otros, y tienen cierto grado de control sobre sus acciones y su estado interno;

- **Habilidad social:** los agentes interactúan con otros agentes (y posiblemente con humanos) vía algún tipo de lenguaje de comunicación entre agentes;

- **Reactividad:** los agentes perciben su ambiente, (que puede ser el mundo físico, un usuario vía una interfaz gráfica, una colección de otros agentes, la INTERNET, o tal vez todos estos combinados), y responden a cambios que ocurren en él;

- **Pro-actividad:** los agentes no actúan simplemente en respuesta a su ambiente, son capaces de exhibir comportamiento oportunista, dirigido por objetivos, tomando iniciativas cuando sea apropiado.

- **Definición fuerte:** un agente, además de las características anteriores tiene una o más de las siguientes características:

- **Nociones mentales:** un agente tiene creencias, deseos e intenciones.

- **Racionalidad:** realiza acciones a fin de lograr objetivos

- **Adaptabilidad o aprendizaje**

- **Veracidad:** un agente no es capaz de comunicar información falsa de propósito.

Un agente fuerte también es llamado “reflectivo” ya que reflexiona sobre su comportamiento en lugar de simplemente reaccionar a estímulos o cambios.

Se requiere en primer lugar identificar las entidades (agentes, objetos y usuarios humanos) que participan en el dominio del problema. Una vez identificadas las clases presentes en el dominio de la aplicación se deben modelar las relaciones que existen entre ellas., existen básicamente dos tipos de relaciones: estáticas y dinámicas.

Las relaciones estáticas presentan aspectos estructurales y evidencia la arquitectura del sistema en términos de agentes, objetos y usuarios humanos y la relación estática entre estos, incluyéndose

formalismos y técnicas que permiten la especificación de relaciones de jerarquía (herencia), dependencia semántica (asociación) y parte-de (agregación).

La relación, de tipo dinámica permite evidenciar la habilidad social como característica de las entidades agentes. Esta habilidad social implica la necesidad de interacción en forma de comunicación entre agentes, y entre agentes y objetos y/o usuarios humanos.

Se reconoce como colaboración, a la interacción en la cual las entidades implicadas son agentes; y coordinación, a aquella en la que interactúan agentes y usuarios humanos. El modelado de las relaciones de colaboración y coordinación constituye la *vista de comunicación*.

Los agentes están situados en un ambiente, y son capaces de percibir este ambiente a través de sensores de algún tipo. Por tanto los agentes deben poseer *información* acerca de su ambiente. por lo cual el marco de especificación de agentes debe ser capaz de representar tanto el estado del ambiente en sí (vista estructural abstracta) como la información que cada agente tiene sobre este ambiente, sus *creencias*.

Un agente autónomo es uno que es capaz y está autorizado a trabajar (razonar, actuar y reaccionar) de forma independiente, en lugar de ser dirigido por otros agentes.

El marco de desarrollo de agentes debe proveer la facilidad de especificar el hecho que los agentes operan de forma activa, sin la directa intervención de los humanos u otros agentes y con cierto grado de dominio sobre su flujo de control.

Los sistemas reactivos a diferencia de los funcionales (sistemas que simplemente reciben alguna entrada, realizan algún tipo de computación sobre ésta y eventualmente producen alguna salida), no finalizan, sino que mantienen una interacción constante con su ambiente, lo que hace imposible utilizar formalismos que empleen pre- y post- condiciones para razonar sobre sus características. El siguiente requerimiento para el marco de especificación de agentes es que debe ser capaz de representar la naturaleza reactiva inherente de los agentes y sistemas multiagentes en general.

Los agentes afectan su ambiente en lugar de dejar en forma pasiva que su ambiente les afecte. En este sentido podríamos decir que los agentes poseen dos tipos de atributos: los atributos de información (creencias y conocimiento) que están relacionados con lo que el agente conoce del mundo que ocupa; y los atributos pro-activos (deseos, intenciones, obligaciones, acuerdos, elecciones, etc.) que son aquellos que en cierta medida guían las acciones del agente.

No existe un consenso claro tanto en la comunidad AI o filosófica acerca de precisamente qué combinación de atributos de información y pro-atributos son más adecuados para caracterizar a los agentes [2].

Un enfoque particularmente interesante que se trata en este trabajo, afirma que son necesarias: creencias, deseos e intenciones (Beliefs, Desires and Intentions - BDI) [3]. Las creencias representan hechos sobre el ambiente del agente, se distinguen de los conocimientos en el sentido que los conocimientos deben ser siempre verdaderos en cambio un agente puede creer algo falso. Los deseos representan los objetivos del agente, que son estados que un agente desea alcanzar, verificar o mantener. Las intenciones otorgan deliberación al agente, podríamos considerar que el comportamiento pro-activo de los agentes está representado en términos de una librería de planes, el agente selecciona un plan de la librería sobre la base de los objetivos que desea satisfacer. Un plan se instancia cuando ocurre el disparo de un evento que satisface su invocación y condiciones de contexto. Un plan instanciado es una intención. El cuerpo de un plan es un conjunto de tareas que pueden ser sub-objetivos, acciones, aserciones de la base de creencias, y consultas y mensajes a otros agentes. Cuando se forma una intención, estas tareas son ejecutadas por el agente en un esfuerzo por alcanzar un objetivo dado.

La racionalidad, por su parte, es la asunción que un agente actuará a fin de lograr sus objetivos, y no lo hará de una manera tal que prevenga que sus objetivos sean alcanzados – al menos en la medida que sus creencias lo permitan. El marco de desarrollo de sistemas basados en agentes debe cubrir el modelado de todas estas características.

Finalmente, cabe destacar que no todos los agentes necesitan tener movilidad. En efecto, un agente puede simplemente comunicarse con su ambiente por medios convencionales. Esto incluye varias formas de llamada a procesos remotos y mensajes [4]. Sin embargo, consideramos la movilidad (que no es una característica presente en la definición de Wooldridge) una característica interesante de los agentes, particularmente de aquellos orientados a Web que constituyen una parte importante de los sistemas existentes y que se encuentran en etapas de rápida evolución.

Un agente móvil no está limitado al sistema donde inicia su ejecución. Es libre de viajar a través de los nodos (hosts) de una red de computadoras. Creado en un ambiente de ejecución, puede transportar su estado y código a otro ambiente de la red, donde reanuda la ejecución. Los agentes móviles pueden ser interesantes porque, entre otras cosas:

- Reducen la sobrecarga de la red: los agentes móviles permiten empaquetar el intercambio de mensajes y enviarlos al destino donde se lleva a cabo la interacción en forma local.
- Encapsulan protocolos: cuando se intercambian datos en un sistema distribuido, cada nodo posee el código que implementa el protocolo necesario para interpretar los mensajes intercambiados. Cuando los protocolos evolucionan, también se necesita actualizar el código que lo implementa. En cambio, los agentes móviles, son capaces de moverse a los nodos estableciendo canales de comunicación basados en protocolos propios.

## 2.2 MaSE

### 2.2.1 Introducción

La cátedra de robótica de la Universidad de Morón llevo adelante una investigación de la metodología en cuestión [25], se presenta a continuación el resultado obtenido. La metodología MaSE toma las especificaciones iniciales de un sistema y produce un conjunto de documentos basados en un estilo gráfico. El principal objetivo de MaSE es guiar a un desarrollador a través de un ciclo de vida del software, desde una especificación escrita en prosa hasta un sistema de agentes implementado.

Las distintas fases de MaSE pueden verse en la figura 1. La salida de cada una de estas fases sirve como entrada a la siguiente fase. Se puede ver que es una metodología de tipo iterativa y con sucesivas repeticiones del ciclo se puede lograr un mejor nivel de detalle en las especificaciones del proyecto.

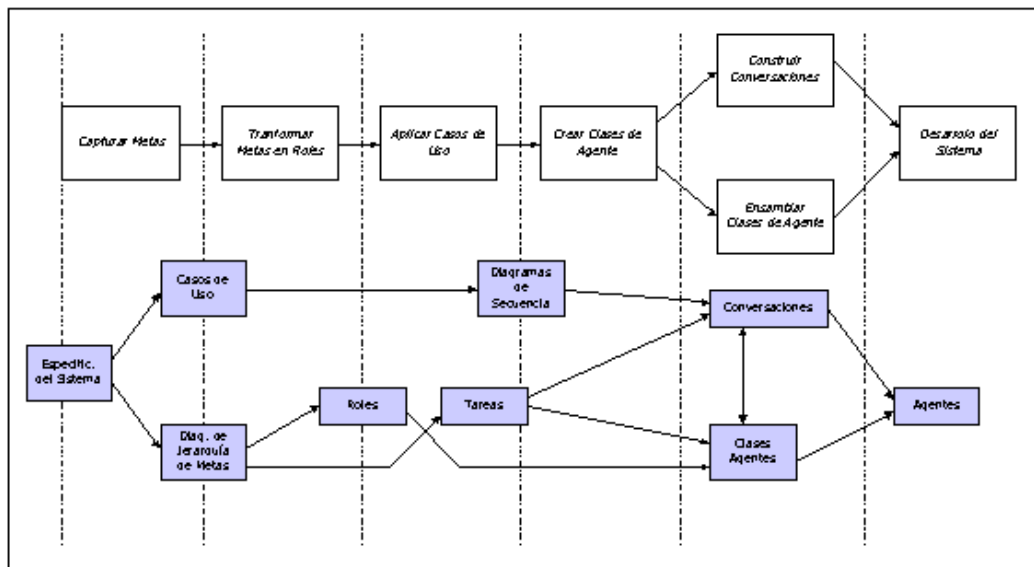


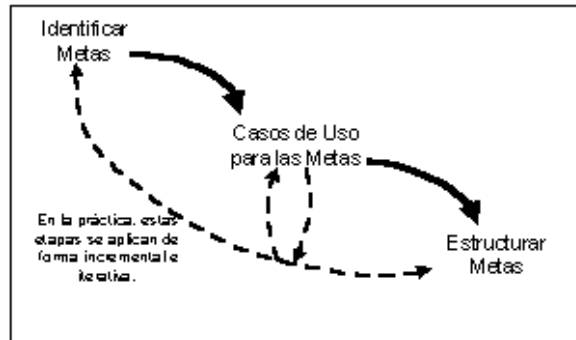
Fig 2.1 Metodología MaSE

### 2.2.2 Capturar metas

Es la primera fase en la metodología, que toma las especificaciones iniciales del sistema y las transforma en un conjunto *metas del sistema*. Para esta fase del ciclo de vida tendremos en cuenta lo investigado por Kendall [5], y el reporte técnico de DeLoach y Wood [6].

En MaSE una meta siempre es definida a nivel del sistema como un objetivo. Como consecuencia, cada acción dentro de un sistema debe soportar una meta determinada. Una meta es una sentencia que generalmente se expresa como una posible funcionalidad del sistema.

Existen tres etapas en la fase *capturar metas*: identificar metas, determinar casos de uso para las metas, estructurar las metas. La relación entre las etapas es que el resultado o salida obtenida en cada etapa servirá como entrada en la etapa siguiente. La figura 2 muestra las etapas de la fase *Capturar Metas*.



**Fig 2.2 Fase Capturar Metas**

La primera etapa, identificar metas, comienza con la extracción de los escenarios<sup>1</sup> de las especificaciones de los requerimientos, historias de usuarios, documentos técnicos detallados, etc. Una vez identificados los escenarios, la manera de establecer cuales son las metas es preguntar “¿Cuál es el objetivo de este escenario?”. Respondiendo esta pregunta podemos decir que, las metas se identifican determinando cual es el propósito de cada escenario. Las metas deben ser especificadas de la manera más abstracta posible sin perder la esencia de los requerimientos.

Luego, en la fase determinar casos de uso para las metas, se definen casos de uso<sup>2</sup> en una narrativa que exprese el flujo de eventos que definen el comportamiento del sistema. Se puede decir que son ejemplos de cómo el usuario, o el autor de los requerimientos, piensa cómo el sistema debería comportarse. Puede que en esta etapa se clarifique cierta información existente sobre las metas del sistema; dado el caso el analista deberá volver y plantear o modificar nuevas metas que luego serán plasmadas en el Diagrama de Jerarquía de Metas. Una manera fácil de identificar Casos de Uso es identificando aquella secuencia de eventos que difiere de otra secuencia de manera significativa. También es importante que el analista capture los casos de uso *positivos* y *negativos*. Los *positivos* son aquellos que representan lo que debe pasar durante la operación normal del sistema, mientras que los *negativos* son aquellos que también describen una secuencia de eventos deseados pero para cuando surgen errores o fallas generales. El objetivo de crear Casos de Uso es identificar posibles canales de comunicación, y no definir todas las posibles combinaciones entre eventos y datos en el sistema.

Finalmente, una vez que las metas son analizadas, se deberán estructurar en lo que se llama Diagrama Jerárquico de Metas. Las metas deben ser descompuestas en pequeñas sub-metas hasta el punto en que parezca que cada sub-meta puede ser manejada por un simple agente. El primer paso es identificar la meta general del sistema, la cual es colocada en la parte superior del Diagrama de Jerarquía de Metas. Las metas en este diagrama son organizadas por importancia. Cada nivel de jerarquía contiene metas que son iguales en importancia y todas las sub-metas se relacionan funcionalmente con su padre de nivel inmediato superior. Cada sub-meta debe relatar las mismas funciones o modos de operación que sus padres, pero en un nivel más bajo.

## 2.2.3 Transformar metas en roles

El segundo paso de la metodología MaSE es transformar las metas que fueron estructuradas en una forma más útil para la construcción de sistemas MultiAgentes: los roles y sus tareas asociadas. Lo que se desarrollará a continuación será en base a lo publicado por Wooldridge, Jennings y Kinny[7] y por Depke, Heckel y Küster [8].

Los roles son los bloques de construcción utilizados para definir clases de agentes durante la fase de diseño. Se puede asegurar que cada una de las metas están asociados con un rol en particular, y además que cada rol es representado o ejecutado por una clase de agente.

Los roles son las bases bajo las cuales los agentes son diseñados.

<sup>1</sup> “Una secuencia específica de acciones que demuestran un comportamiento. Un escenario puede ser utilizado para mostrar una interacción o la ejecución de un caso de uso.” *OMG Unified Modeling Language Specification (draft), Febrero de 2001.*

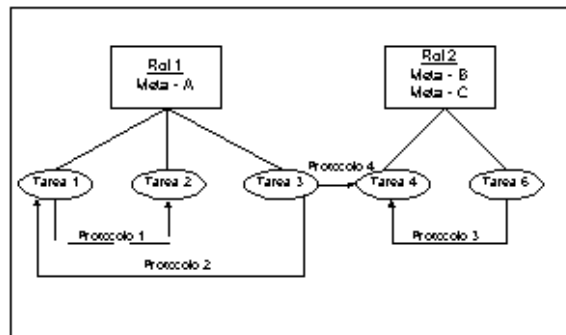
<sup>2</sup> “Un caso de uso especifica los conceptos utilizados para la definición de la funcionalidad de una entidad como puede ser un sistema o subsistema, sin especificar su estructura interna.” *OMG Unified Modeling Language Specification (draft), Febrero de 2001.*

Los roles están definidos por cuatro atributos: responsabilidades, permisos, actividades y protocolos:

- 1) Responsabilidades: un rol es creado para hacer algo. Esto es, un rol tiene cierta funcionalidad. Esta funcionalidad es representada por este atributo, y tal vez sean la clave para asociar a los roles. Las responsabilidades se dividen en dos:
  - a. Propiedades de vida: generalmente indican que “algo bueno pasa”. Describen aquellos estados por los que un agente debe pasar dadas ciertas condiciones ambientales. Indican que el agente que está representando o ejecutando un rol está todavía con vida
  - b. Propiedades de seguridad: son invariantes. Indican que “nada malo pasa”.
- 2) Permisos: son los derechos asociados a los roles. Identifican los recursos que están disponibles para un rol para la realización de sus responsabilidades. Generalmente tienden a ser recursos de información. Los permisos tienen dos aspectos a tener en cuenta:
  - a. Identifican los recursos que pueden ser utilizados legítimamente para llevar a cabo un rol.
  - b. Indican el límite de recursos que un ejecutor de un rol puede operar.
- 3) Actividades: son procedimientos asociados con los roles que pueden llevarse a cabo por un agente sin necesidad de interactuar con otros agentes. De esta manera se puede decir que son acciones privadas.
- 4) Protocolos: definen la manera en que los roles pueden interactuar con otros roles.

El caso general de transformación de metas a roles es que una meta se asocia únicamente con un rol. Sin embargo se puede dar el caso en que más de una meta se pueda combinar y así asignar este conjunto de metas a un rol en particular por cuestiones de eficiencia y teniendo en cuenta las propiedades de cohesión. Pressman [9].

A continuación se describe la manera de documentar a los roles en la metodología MaSE. Primero, se listan las metas asociadas a cada rol. Los roles son representados por rectángulos, mientras que las tareas mediante óvalos pegados a los roles. La siguiente es una figura que ilustra la manera de documentar roles.



**Fig 2.3 MaSE Modelado de Roles**

Las líneas que unen las distintas tareas representan protocolos de comunicación entre las tareas. Las flechas se corresponden con la relación iniciador / contestador del protocolo, saliendo desde el iniciador hasta el contestador. Las líneas sólidas indican comunicaciones que generalmente se implementan como protocolos externos de comunicación. Los protocolos externos incluyen el pasaje de mensajes entre roles que pueden llegar a ser mensajes propios si los roles se implementan sobre agentes separados. Las líneas punteadas denotan comunicación entre tareas concurrentes dentro del mismo rol. Los roles no tienen la posibilidad de compartir o duplicar tareas. El hecho de compartir una tarea significa una mala descomposición de roles. Las tareas compartidas deben ser situadas en roles distintos, que puedan ser combinadas en diferentes clases de agentes en la fase de Diseño.

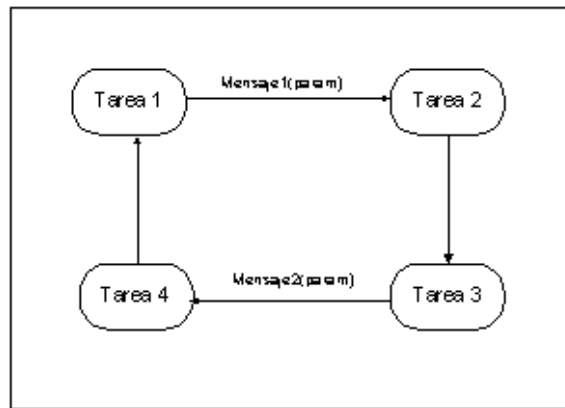
## 2.2.4 Aplicar casos de uso

Luego de que los roles fueron creados, se asocian tareas para cada rol. Cada meta asociada con un rol puede tener tareas que definan los detalles de cómo se logra o alcanza una de las metas planteadas. Esto se debe hacer luego de la creación de los roles ya que las tareas se comunican con otras tareas en varios roles. Wood y DeLoach [10].

El comportamiento de un rol lo especificamos mediante un conjunto de  $n$  tareas concurrentes. Cada tarea especifica un único hilo de control que define un comportamiento particular que el rol debe mostrar.

Las tareas concurrentes se especifican gráficamente a través de una máquina de estados<sup>3</sup> finitos, al que se le llama Diagrama de Tareas Concurrentes. Se asume que todas las tareas comienzan su ejecución al principio de ejecución de un rol y continúan hasta que el rol termina o llega a algún estado de finalización.

Las tareas se representan a través de rectángulos con vértices circulares, mientras que el paso de mensajes entre tareas se representa mediante una flecha con el nombre del mensaje por encima de la misma. La siguiente es una figura de diagrama de tareas concurrentes.



**Fig 2.4 MaSE Modelado Tareas**

Otros de los diagramas a tener en cuenta para realizar en esta fase de la metodología es el Diagrama de Secuencia<sup>4</sup>. Se toman los casos de uso identificados en la fase Capturar Metas y se reestructuran como diagramas de secuencia.

El Diagrama de Secuencia se utiliza para determinar la cantidad mínima de mensajes que se debe pasar entre roles. Si se pasa un mensaje entre roles, entonces debe existir algún camino de comunicación entre ellos. Un camino de estos entre roles interpretados por diferentes clases *agente*, indica que debe existir también una conversación entre dos clases *agente* para pasar el mensaje. La clase que interpreta el rol que inicia la comunicación se llama *iniciador* de la conversación y la clase que recibe *contestador*.

Por lo general se construye un diagrama de secuencia por cada caso de uso identificado.

<sup>3</sup> "Comportamiento que especifica la secuencia de estados que un objeto o una interacción atraviesa durante su vida en respuesta a los eventos, junto con sus respuestas y acciones." *OMG Unified Modeling Language Specification (draft). Febrero de 2001.*

<sup>4</sup> "Un diagrama que muestra interacciones entre objetos organizado en una secuencia de tiempo. En particular, muestra los objetos participando en la interacción y la secuencia de mensajes que se intercambian." *OMG Unified Modeling Language Specification (draft). Febrero de 2001.*

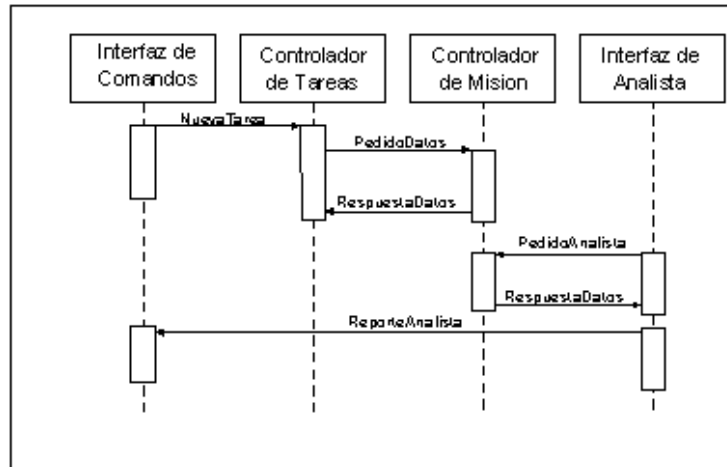


Fig 2.5 MaSE Diagrama de Secuencia

## 2.2.5 Crear clases Agente

En esta fase de la metodología, las clases agentes se identifican a partir de los roles. El producto de esta etapa de la metodología es el *Diagrama de Clases de Agentes*, que muestra las clases agente existente junto con las conversaciones que mantienen dichos agentes. “Una clase agentes es una plantilla para un determinado tipo de agente que habrá en el sistema, de la misma manera que una clase objeto es una plantilla para el paradigma de objetos.” Wood [11]. La principal diferencia entre un diagrama de este tipo y su similar en el paradigma de objetos son las relaciones que existen entre los agentes, que para este caso se llaman *conversaciones*.

Cada rol puede ser interpretado por una clase agente, y una clase agente puede interpretar uno o mas roles cambiando entre ellos de manera dinámica. Además agentes de la misma clase agente pueden interpretar diferentes roles al mismo tiempo.

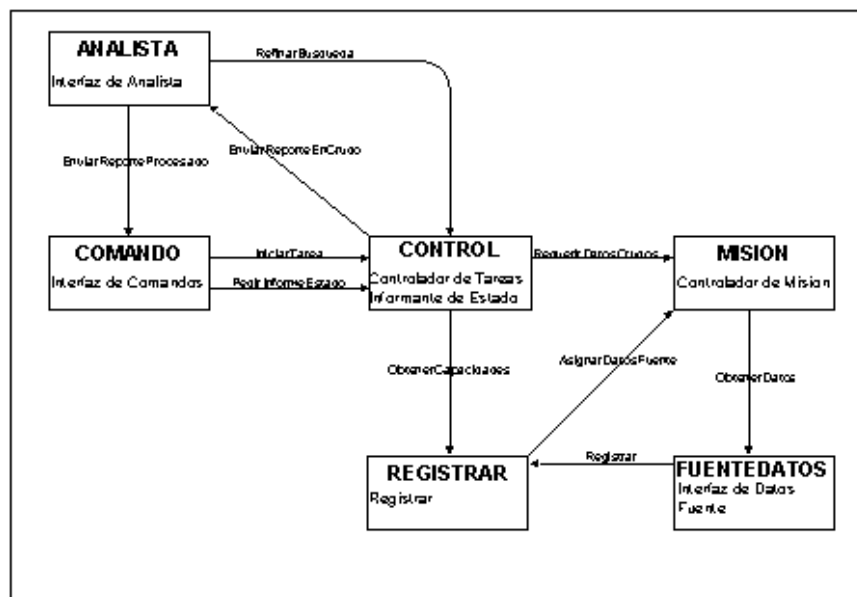


Fig 2.6 MaSE Diagrama de Clases Agente

## 2.2.6 Detalles de diseño

De la misma que se mencionó en, Transformar Metas en Roles, existe un mapeo de uno-a-uno con respecto a roles y clases agente. Esto es, generalmente por cada rol se crea una clase agente. Por

supuesto, que en la experiencia y conveniencia del diseñador juntar varios roles y mapearlos en una misma clase agente. Debido a que los agentes heredan los caminos de comunicación mencionados anteriormente para los roles, estos se convierten en conversaciones para las clases agente. En el momento de determinar que roles se van a combinar, es necesario tener en cuenta cual va a ser la frecuencia y el tamaño de la comunicación. También es importante identificar si se requieren clases agente para hacer de interfaz con recursos externos, como por ejemplos humanos, otros sistemas de software, etc.

## 2.2.7 Construir conversaciones

Esta es la siguiente fase en la metodología MaSE. Una conversación bajo MaSE define el protocolo coordinado entre dos agentes. La metodología define específicamente una conversación como dos *Diagramas de Comunicación de Clases*, uno para el iniciador de la conversación y otro para el que responde. Este diagrama es un par de máquinas de estados finitos que definen los estados de la conversación de las dos clases agentes que participan en la misma.

La siguiente figura muestra la máquina de estados finitos para la parte iniciadora de una conversación.

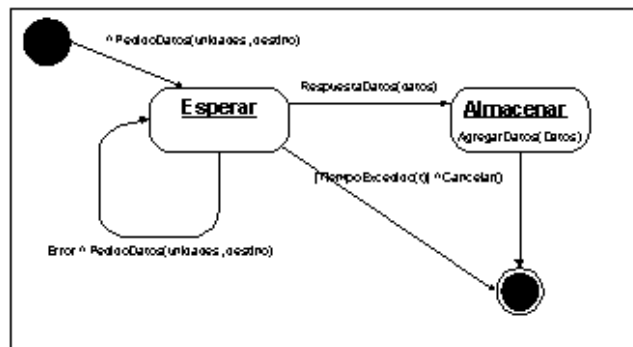


Fig 2.7 MaSE Diagrama de Comunicación de Clases – Iniciador -

La sintaxis de las transiciones (flechas que van de una estado a otro) siguen la notación convencional de UML que se muestra a continuación:

$$rec-mes(args1) [condición] \wedge trans-mes(args2)$$

Esta notación indica que, si un mensaje *rec-mes* se recibe con argumentos, *args1*, y la condición, *condición*, se mantiene, entonces el mensaje *trans-mes*, se transmite con los argumentos *args2*. La conversación pasa del estado que comienza hasta el estado al que apunta la flecha de transición. Continúa de esta manera hasta que la transición alcanza un estado de finalización.

Cuando un agente recibe un mensaje, lo compara contra sus conversaciones activas en ese momento. Si encuentra alguna coincidencia, el agente hace una transición de la conversación apropiada a un nuevo estado y realiza cualquier actividad requerida por la transición o el nuevo estado. De lo contrario, el agente compara el mensaje con todas las conversaciones en las que puede participar con el agente que envió el mensaje, y establece una nueva conversación si el mensaje coincide con alguna transición que salga desde el estado de comienzo. Cualquier actividad que ocurra durante una transición o en un estado, deberán ser mapeadas a algún método en su correspondiente clase agente.

## 2.2.8 Ensamblado de agentes

En esta etapa de MaSE se construye la parte interna de una clase agente. En este apartado trataremos de dar una noción general de la investigación de Robinson [12].

Se describirá el proceso de selección para los distintos tipos de arquitectura. Se definen cuatro estilos de arquitectura a saber: *reactiva*, *basada en conocimiento*, *planeamiento*, *BDI (Belief, Desire, Intention)*.

## 2.2.9 Arquitectura de Agente Reactivo.

Debido a la naturaleza de estímulo - respuesta de los agentes reactivos, todas las arquitecturas de esta categoría están basadas en un conjunto de reglas. Estas reglas se utilizan para interpretar los estímulos y generar una respuesta si es necesario. Básicamente la estructura utilizada aquí es la sentencia de condición IF.... THEN.

Para recibir señales y emitir sus respuestas, cada arquitectura contiene una interfase para interactuar con el ambiente en el que se encuentra. Estas interfaces incluyen una interfaz de mensajes, recursos, sensores y efectores. La interfaz de mensajes se utiliza para enviar y recibir mensajes con otros agentes. Los sensores se utilizan para percibir las variables que le interesan al agente de su ambiente, y los efectores se utilizan para producir cambios en el ambiente. Por último la interfaz de recursos se utiliza para intercambiar información con estructuras externas como por ejemplo bases de datos.

Basado en UML, Robinson define un tipo de notación de más alto nivel, permitiendo la interpretación del estilo de arquitectura de una manera más fácil. La siguiente figura muestra esta notación para la arquitectura de Agentes Reactivos.

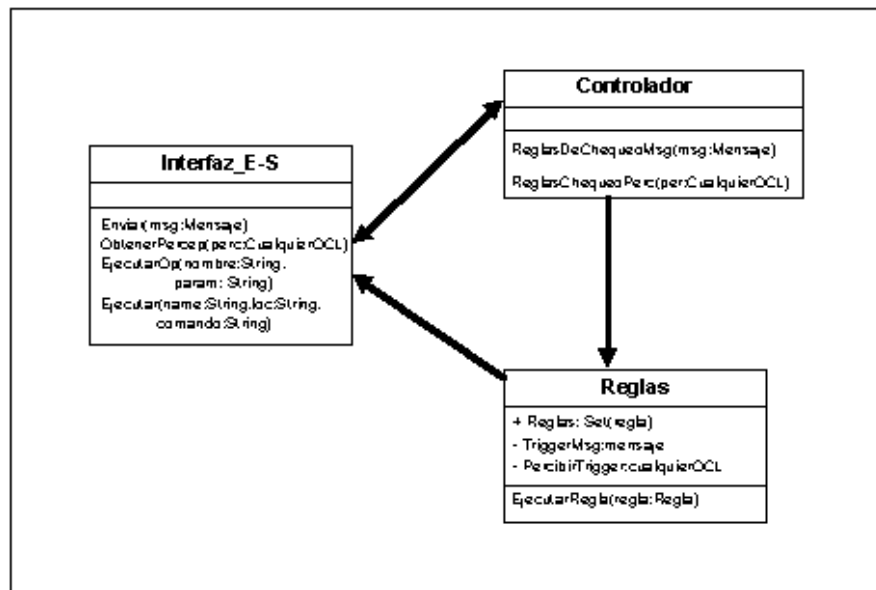


Fig 2.8 Diagrama de Componentes para Agentes de Arquitectura Reactiva

A su vez, se puede crear un diagrama de clases (mas bien de objetos) y que muestra en detalle la arquitectura de los agentes reactivos. La siguiente figura muestra el diagrama. Este diagrama sólo se mostrará a modo de ejemplo para los agentes reactivos, obviándolo para las demás arquitecturas.

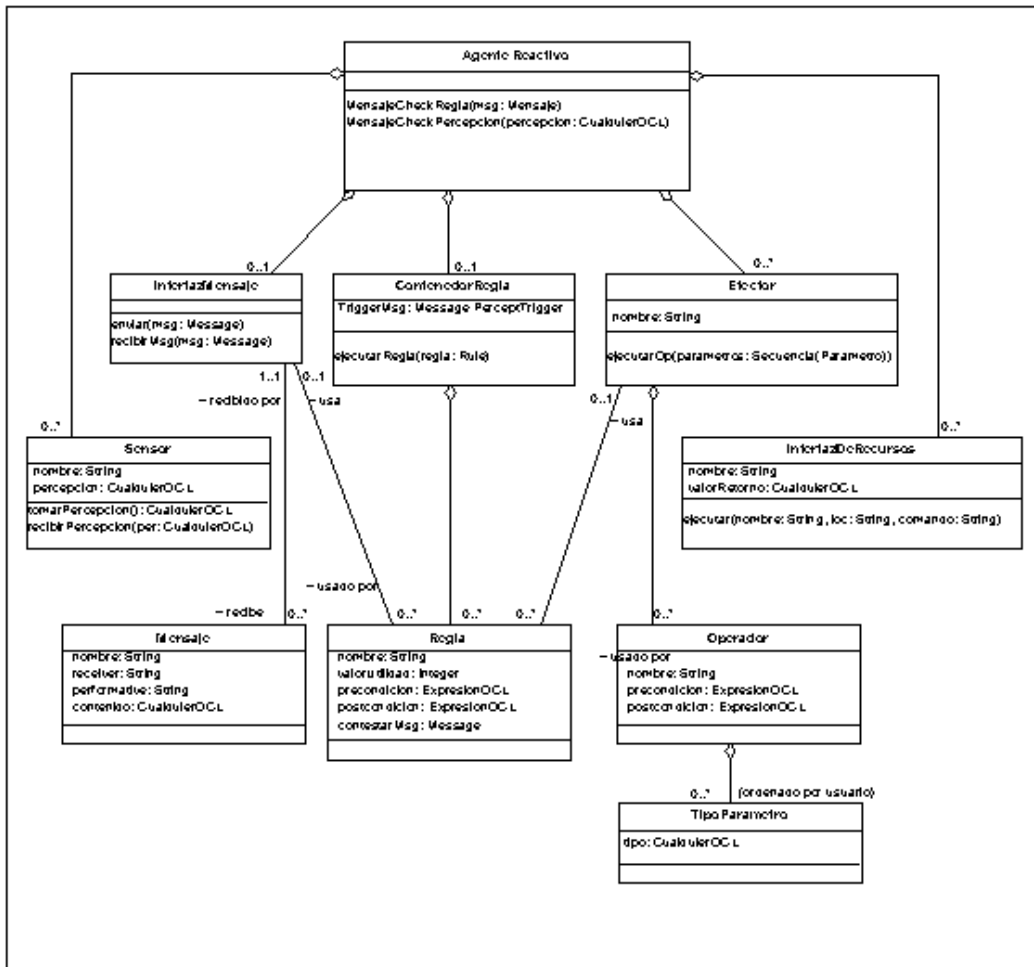


Fig 2.9 Diagrama de Clases para Agentes de Arquitectura Reactiva

## 2.2.10 Arquitectura de Agente basado en Conocimientos

Como el nombre lo indica este tipo de arquitectura está basada en la base de conocimientos del agente. La base tiene información predefinida que el agente puede necesitar, como así también cualquier información obtenida o aprendida por el agente. De la misma manera que en la arquitectura reactiva, las arquitecturas basadas en conocimiento deben seguir un conjunto de reglas para tomar las decisiones.

Aunque se considera a la base de conocimiento como el elemento fundamental de este tipo de arquitectura, también consideramos al motor de inferencia como el segundo elemento más importante de la arquitectura. Este motor provee de facilidades al agente y la posibilidad de navegar a través de la información, contenida en la base de conocimientos, con el objetivo de deducir algo por medio de esa información. El motor utilizará reglas y la base de conocimiento para razonar sobre la información y deducir resultados de una manera organizada.

Debido a que este tipo de arquitectura tiene la habilidad de aceptar pedido y actualizaciones externos de otras fuentes también se incluirá una interfase de entrada – salida. “Luego de revisar varias arquitecturas basadas en conocimiento, los componentes principales de este tipo de arquitectura son: interfaz entrada – salida, contenedor de reglas, controlador, base de conocimientos y motor de inferencia.” Robinson [12]. La siguiente figura muestra el diagrama de componentes.

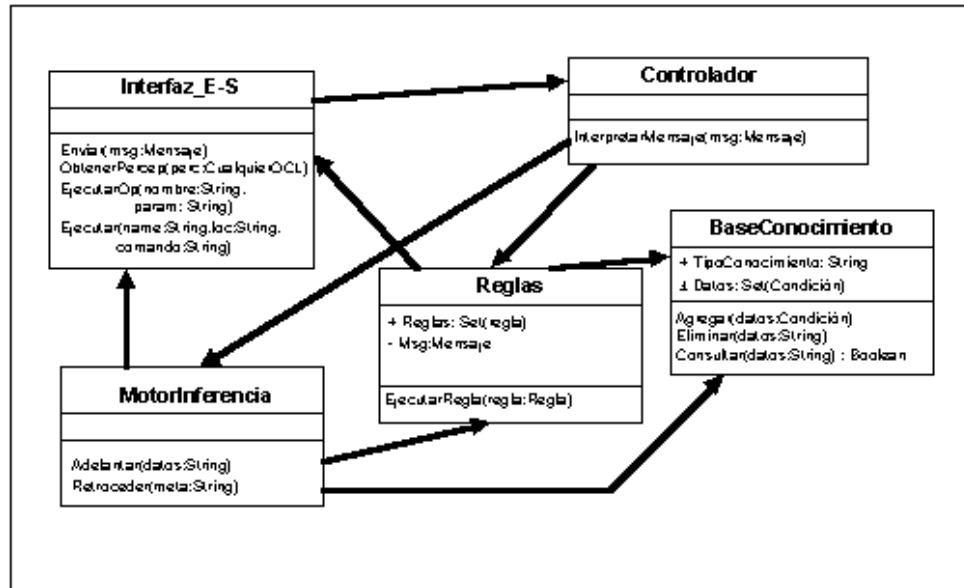
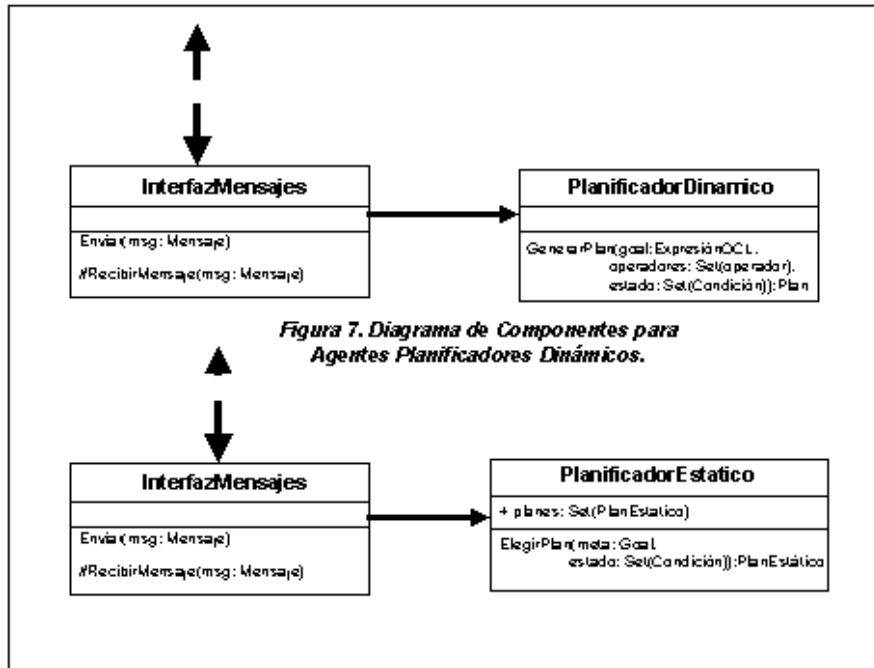


Fig 2.10 Diagrama de Componentes para Agentes Basados en Conocimientos

### 2.2.11 Arquitectura de Planeamiento

Existen dos tipos de planificadores para cualquier sistema de Inteligencia Artificial: planificadores dinámicos y planificadores estáticos. Los planificadores dinámicos toman como entrada una meta, un conjunto de operadores, y el estado del agente, y de forma dinámica generan un plan para satisfacer la meta del agente. Los planificadores estáticos toman como entrada la meta y estado del agente, y utilizan esta información para elegir el plan existente que mejor satisfaga la meta. Aunque la manera de ejecución es totalmente diferente, la estructura básica de estos planificadores es la misma. Ambos poseen una interfase de entrada – salida para recibir los mensajes de otros agentes. Los mensajes tienen la información necesaria para elegir o crear el plan de ejecución. Debido a que lo único que debe hacer el agente es generar o elegir un plan, no es necesaria ninguna otra interfaz.

En general, un plan consiste en una secuencia de uno o más pasos, cada uno de los cuales tiene un conjunto de atributos. Cada paso tiene el nombre del operador que debe ejecutarse y una obligación. La obligación es una lista de parámetros que el agente debe pasar. Con lo mencionado el diagrama de componentes de este tipo de arquitectura es el que muestra la siguiente figura.



**Fig 2.11 Diagrama de Componentes para Agentes Planificadores Estáticos**

### 2.2.12 Arquitectura BDI

Los atributos que lógicamente tiene en cuenta este tipo de arquitectura son: Creencias, Deseos e Intenciones.

Un agente puede tener distintos tipos de creencias, orientadas hacia el ambiente en el que se desenvuelve o también hacia otros agentes. Los deseos se corresponden con las metas que el agente debe satisfacer. Las metas pueden ser elegidas por el usuario, o bien el agente puede adoptarlas. Las intenciones de un agente son simples de representar, debido a que es una lista de planes que el mismo intentará alcanzar. Para que un agente logre cualquiera de sus deseos, se debe elegir un plan existe o generar uno que satisfaga la meta. De esta manera una agente BDI, debe usar planificadores dinámico, estáticos o ambos a la vez.

Las interfaces utilizadas en este tipo de arquitectura varían según la meta general que debe conseguir el agente, pero todas incluyen sensores, efectores, interfaces de mensajes y de recursos. Se utiliza un controlador centralizado para dirigir la información, como también para controlar la ejecución de las intenciones del agente. La siguiente figura muestra los componentes de este tipo de arquitectura.

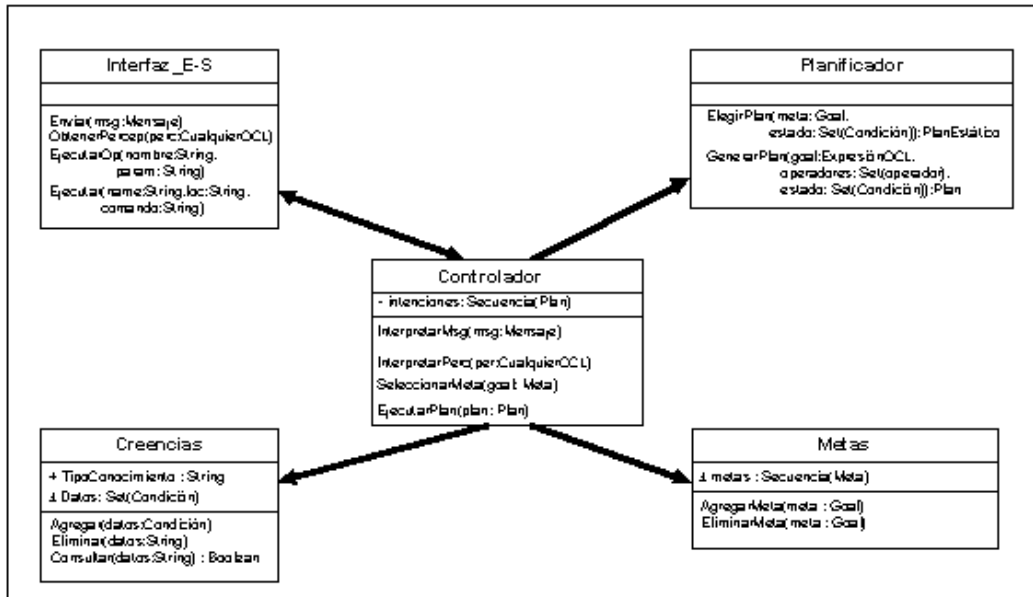
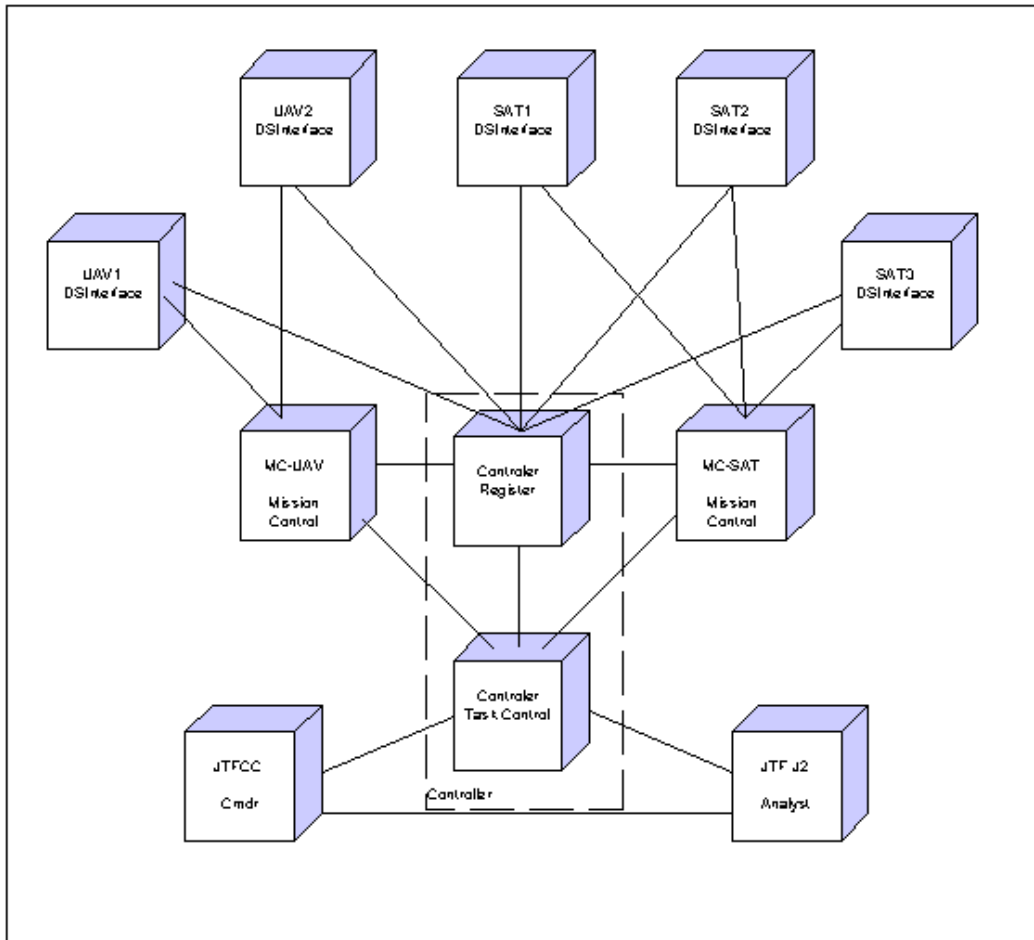


Fig 2.12 Diagrama de Componentes para Agentes de Arquitectura BDI

### 2.2.13 Desarrollo del sistema

La fase final de la metodología MaSE toma las clases agente y las instancia como agentes reales. Se utiliza un diagrama de desarrollo para mostrar números, tipos y localidades de los agentes dentro del sistema. Este diagrama es el más simple de la metodología, ya que la mayor parte del trabajo se hizo en fases anteriores.

El objetivo del *Diagrama de Desarrollo* es definir un sistema basado en clases agentes definidas en las fases previas de MaSE. La figura siguiente muestra un Diagrama de Desarrollo.



**Figura 2.13 Diagrama de Desarrollo**

Las cajas de tres dimensiones representan a los agentes y las líneas de conexión las conversaciones entre los mismos. Un rectángulo punteado indica que varios agentes pueden estar en la misma plataforma física.

Si bien en la fase de MaSE de ensamblado de agentes no se definen claramente los diagramas de clases u objetos pertenecientes a los distintos tipos de arquitecturas de agentes, y por otro lado se muestra solamente un ejemplo de cómo serían los mismos, se anexa una investigación o, mejor dicho, una proposición hecha por Giret, Adriana, Cernuzzi, Luca, Pastor, Oscar y Insfrán, Emilio [13].

Particularmente se detalla el diseño de agentes, en lo que se refiere a diagramas de objetos y clases.

Para la especificación de agentes son necesarias tres vistas o modelos:

- 1) Vista Estructural Abstracta:
  - a. Identificación de agentes: Los agentes son las entidades activas del sistema, que pueden cambiar su propio estado y cuyas acciones pueden modificar el ambiente; el ambiente por su parte, consiste en elementos pasivos (objetos) cuyos estado cambian solo por las acciones de los agentes o de los actores.
  - b. Modelo de Objetos: Una clase agente, se representará gráficamente como una clase, con el estereotipo <<agente>> en la cabecera. Se podrán establecer relaciones (agregación, herencia) convencionales de las metodologías orientadas a objetos.

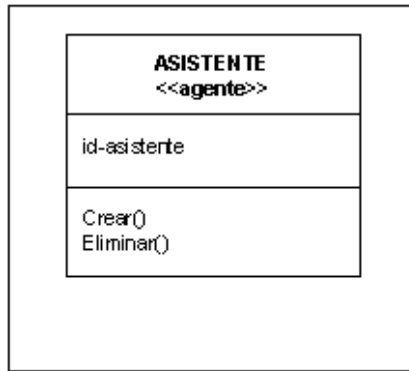


Figura 2.14 Notación Gráfica para una Clase Agente

- 2) Vista de Comunicación:
- a. Modelo Dinámico: en el diagrama de interacción las clases agentes será representadas en los disparos e interacciones globales por cajas con la etiqueta <<agente>>.

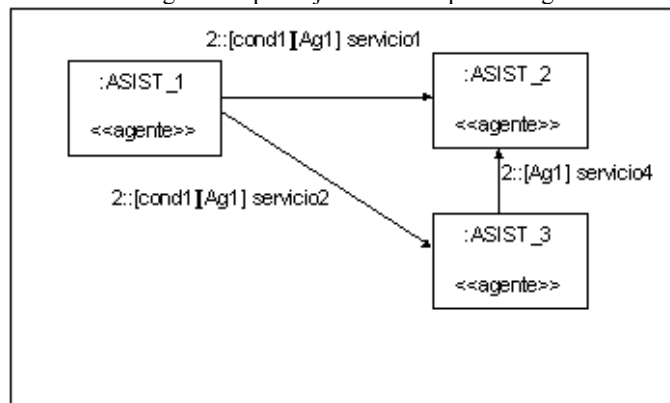


Figura 2.15 Notación Gráfica coordinación entre Agentes

- 3) Vista Interna de cada Agente: Modelo de Objetos: por cada clase agente declarada se proveerá la definición de las nociones mentales (creencias, deseo e intenciones) del agente.
- a. *Creencias*. serán modeladas por medio de clases componentes de la clase agente. Los atributos  $a1, \dots, an$  representarán las creencias sobre los objetos del entorno, mientras que las operaciones  $s1, \dots, sn$  corresponderán a los métodos para el manejo de las creencias.



Figura 2.16 Clase Creencias del Agente

- b. *Deseos u Objetivos*: serán modelados por medio de clases componentes de la clase agente. El atributo *objetivo* será el objetivo propiamente dicho.

- c. *Intenciones*: un plan instanciado es una intención, por lo tanto en un modelo conceptual es preciso modelar para cada objetivo cuáles serán los planes del agente que podrán ser sus intenciones. Los planes serán clases componentes de la clase Objetivos. Por cada objetivo podrá existir mas de un plan. El cuerpo de un plan será modelado como una transacción global, que será ofertada como un servicio de la clase planes. La ejecución de un plan específico está determinada por el valor de verdad de la condición asociada al plan. Este hecho será modelado por medio de disparos que ejecutan la transacción, según la condición asociada.

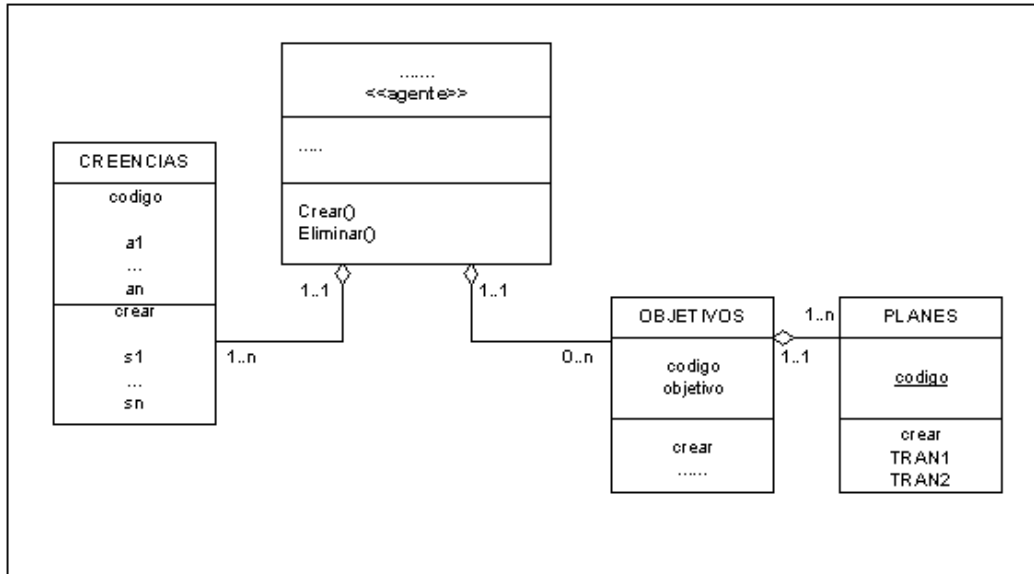


Figura 2.17 Estructura de las Nociones Mentales de una Clase Agente

## 2.3 Resumen de Metodologías Orientadas a Agentes

La mayoría de las metodologías orientadas a agentes tienen un sustento teórico basado en metodologías basadas en objetos. Pero a su vez existen algunas que tienen sustentos teóricos de Ingeniería de Conocimiento. A continuación se hará una síntesis de algunas de las metodología existentes: Cernuzzi y Giret [14]

### 2.3.1 Análisis y Diseño Orientado a agentes

Esta técnica (Burmeister 1996) define tres modelos para el análisis de sistemas basados en agentes: el modelo *agente*, que contiene a los agentes y su estructura interna (creencias, planes y metas); el modelo *organizacional*, que describe las relaciones entre los agentes (herencia y roles en la organización); y el modelo de *cooperación*, que describe las interacciones entre los agentes.

### 2.3.2 Técnica de Modelado de Agentes para sistemas de agentes BDI

BDI define dos niveles principales (interno y externo) para los agentes BDI (Belief, Desire and Intention)

El punto de vista externo consiste de una descomposición del sistema en agentes y la definición de sus interacciones. Esto se ve a través de dos modelos: el *modelo de agente*, para describir las relaciones de jerarquías entre los agentes; y el *modelo de interacción*, para describir las responsabilidades, servicios e interacciones entre los agentes y sistemas externos.

El punto de vista interno se lleva a cabo a través de tres modelos: el *modelo de creencias*, que describe las creencias sobre el ambiente; el *modelo de metas*, que describe las metas y eventos que un agente puede adoptar o responder; y el *modelo de planes*, que describe los planes que un agente puede utilizar para conseguir sus metas.

### 2.3.3 Método basado en escenario para sistemas MultiAgentes (MASB)

Este método (Moulin y Cloutier 1994) es aplicado para sistemas multiagente en el campo de trabajo cooperativo.

La fase de análisis consiste de las siguientes actividades: descripción de escenarios, descripción funcional de roles, modelado conceptual del mundo y datos, y modelo de interacción sistema-usuario.

La fase de diseño consiste de: arquitectura MAS y descripción de escenario, modelado de objetos, modelado de agentes, modelado de conversaciones y validación total del sistema de diseño.

### 2.3.4 Metodología CoMoMAS

Esta es una extensión a la metodología CommonKADS (Schreiber 1994) para el modelado MAS. Se definen los siguientes modelos:

- 1) Modelo de Agente: define la arquitectura de los agentes y el conocimiento de los mismos. Se clasifican como conocimientos sociales, cooperativos, de control, cognitivos y reactivos.
- 2) Modelo Expertise: describe la competencia reactiva y cognoscitiva del agente.
- 3) Modelos de Tarea: describe la descomposición de tareas y detalles si la tarea fue resuelta por un usuario o agente.
- 4) Modelo de Cooperación: describe la cooperación entre agentes.
- 5) Modelo de Sistema: define los aspectos organizacionales de la sociedad del agente conjuntamente con los aspectos arquitectónicos.
- 6) Modelos de Diseño: recoge los modelos previos con el objeto de hacerlos operacionales junto con los requerimientos no funcionales.

### 2.3.5 Metodología MAS-CommonKADS

Esta metodología [15] comienza con una fase de conceptualización que es una fase informal que se utiliza para recolectar los requerimientos de usuario y se obtiene una primera descripción del sistema desde el punto de vista del usuario.

La metodología define los siguientes modelos:

- 1) Modelo Agente: describe las características principales de los agentes, incluyendo la capacidad de razonamiento, servicios, metas, etc.
- 2) Modelo de Tarea: describe las tareas (metas) que ejecuta un agente y una descomposición de las mismas.
- 3) Modelo Expertise: describe el conocimiento necesario que necesitan los agentes para llevar a cabo las tareas.
- 4) Modelo de Coordinación: describe las conversaciones entre agentes, esto es, sus interacciones, protocolos y capacidades requeridas.
- 5) Modelo de Comunicación: detalla las interacciones humano-agente a través del software, y los factores humanos para desarrollar estas interfaces.
- 6) Modelo de Diseño: recolecta los modelos anteriores y los subdivide en *diseño de aplicación*, *diseño arquitectónico* y *diseño de plataforma*.

## 2.4 Extensiones de UML para el Modelado de Agentes

### 2.4.1 Diagramas de Secuencia

- 1) Definición: Un Diagrama de Secuencia es una diagrama que muestra la interacción entre objetos en una secuencia de tiempo. En particular, muestra los objetos que participan en la interacción y la secuencia de mensajes que se intercambian.

- 2) Notación: Un diagrama de secuencia tiene dos dimensiones:
- Una vertical que indica el transcurso del tiempo.
  - Una horizontal que representa los diferentes objetos.

Normalmente el tiempo transcurre a medida que avanzamos en el gráfico de arriba hacia abajo. No existe restricción en cuanto a cuál debe ser el eje que se utilice tanto para el transcurso del tiempo como para la identificación de objetos. Se muestra un ejemplo de Diagrama de Secuencia en la Figura 2.18

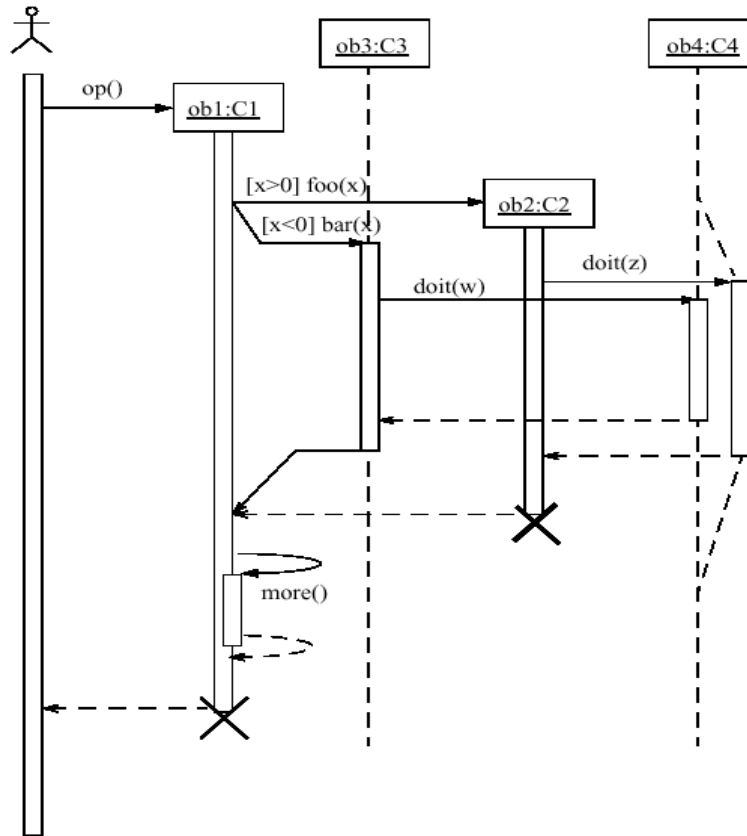


Fig 2.18 Diagrama de Secuencia

### 2.4.2 Línea de Vida de un objeto

En un diagrama de secuencia la línea de vida de un objeto representa una instancia que está interpretando un rol específico. La línea de vida de un objeto se representa a través de una línea punteada. Esta línea representa la existencia de una instancia durante un determinado período de tiempo. Si una instancia se crea o destruye durante un período de tiempo que se muestra en el diagrama, la línea comienza o se detiene en el punto apropiado y determinado por esas operaciones.

Una línea de vida puede separarse en dos o más líneas concurrentes, lo que indica la existencia de una condición. Cada línea se corresponde con cada estado de la condición, y las mismas pueden volverse a juntar en un punto posterior en el tiempo.

### 2.4.3 Activación

Una activación muestra el período de tiempo durante el cual una instancia está ejecutando una acción de manera directa o a través de un procedimiento subordinado. Representa la duración de la acción en tiempo, como también la relación de control entre la activación y sus llamados.

Gráficamente es un rectángulo angosto cuyo principio indica el comienzo en el tiempo de la ejecución de la acción, y cuyo final denota la finalización de la misma en el tiempo. La acción que se está desarrollando puede etiquetarse como un texto cercano a la línea de activación o en su margen izquierdo, dependiendo del estilo que quiera utilizarse.

## 2.4.4 Mensajes y estímulos

Un estímulo es una comunicación entre dos instancias que transportan información con la expectativa que la acción se seguirá. Un estímulo causará que se invoque una operación, se levante una señal, o que cause la creación o destrucción de una instancia.

Un mensaje es una especificación de un estímulo, por ejemplo especifica los roles a los que una instancia para enviar y una instancia para recibir deben adecuarse, como también la acción, cuando se esté ejecutando, despache un estímulo que se adecue al mensaje.

En un diagrama de secuencia, tanto un estímulo como un mensaje se representan mediante una flecha desde la línea de vida de una instancia hasta la línea de vida de otra instancia. En el caso que el estímulo o mensaje se envíe a la misma instancia la flecha deberá comenzar y terminar en la línea de vida de esa instancia.

## 2.4.5 Aplicación de diagramas de secuencia a Agentes

La siguiente figura muestra los elementos básicos para la comunicación entre agentes. El rectángulo puede representar tanto agentes de forma individual como también un conjunto de agentes agrupados. Por ejemplo un agente individual puede etiquetarse *Juan/Cliente*. Aquí Juan es una instancia de un agente que está representando el rol de Cliente. Para indicar que Juan es una Persona – independientemente del rol que esté interpretando- Juan puede expresarse como *Juan:Persona*. La manera básica de etiquetar un agente es *nombre-agente/rol:clase*. La sintaxis mencionada antes ya es parte de UML pero con la diferencia que UML indica el nombre de un objeto en lugar de un agente. Se muestra un ejemplo en la Figura 2.19

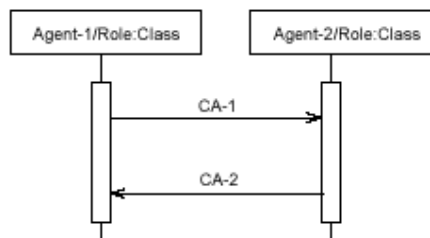


Fig 2.19 Diagrama de Secuencia

Otra recomendación importante de UML es la posibilidad de considerar el soporte para hilos concurrente de interacción. La siguiente figura muestra tres maneras distintas de expresar hilos múltiples. La primera figura indica que todos los hilos CA-1,...,CA-n se envían de manera concurrente. La segunda figura incluye una caja de decisión que decidirá que CAs (cero o más) se enviarán. Si se envía más de una CA la comunicación es concurrente. La figura 2.20 muestra una *or excluyente*, de manera que una sola CA se enviará.

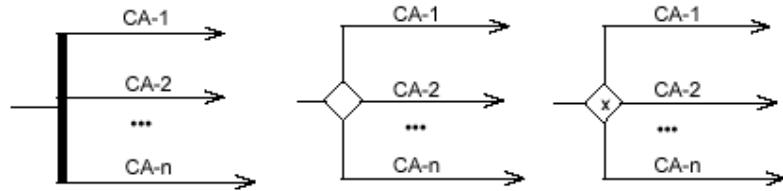


Fig 2.20 Diagrama de Secuencia con OR Excluyente

## 2.4.6 Diagramas de Colaboración

- 1) Definición: Un diagrama de colaboración representa o bien una Colaboración, que contiene un conjunto de roles que serán interpretados por instancias, como también las relaciones necesarias dado un contexto en particular, o bien representa una Interacción que define un conjunto de mensajes (estímulos) especificando la interacción entre instancias que interpretan los roles para lograr resultados esperados.
- 2) Notación: Un Diagrama de Colaboración muestra un gráfico de Instancias unidas a otras. También puede incluir una comunicación establecida por una Interacción. El orden de interacción se describe mediante una secuencia de números, generalmente comenzando con el número 1. Se muestra un ejemplo en la figura 2.21

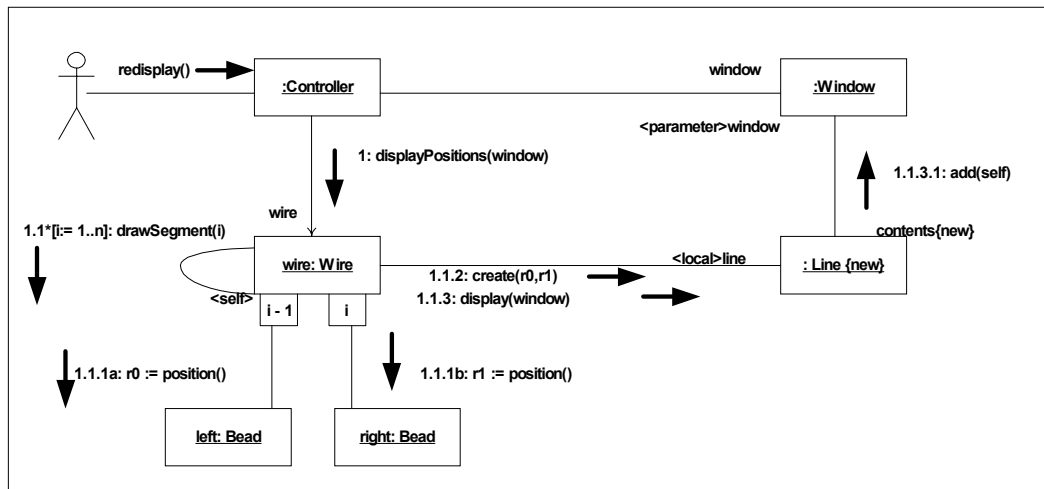


Fig 2.21 Diagrama de Colaboración

- 3) Mensajes y estímulos: En estos diagramas un estímulo es una comunicación entre dos instancias que comparten información esperando que ocurra alguna acción. Un estímulo causará que se invoque una operación, elevar una señal o crear o destruir una instancia.

Un mensaje es una especificación del estímulo, por ejemplo especifica los roles que deberán interpretar el emisor y el receptor del mensaje.

En cuanto a la notación de este tipo de diagrama, lo más importante de destacar es que la unión de las clases presentes en el diagrama, indican que se utilizan para el transporte de los estímulos a la instancia destino.

Algunos flujos de control que se pueden mencionar:

**filled solid arrowhead** →

Puede ser utilizada para llamadas a procedimientos, pero también puede ser utilizada para indicar instancias concurrentes activas cuando una de ellas envía una señal y espera la ocurrencia de una secuencia antes de seguir con su línea de actividad.

**Stick arrowhead** 

Se utiliza para comunicaciones asincrónicas. El transmisor envía el estímulo e inmediatamente continúa con el próximo paso en su ejecución.

**Dashed arrow with stick arrowhead** 

Retorno de una llamada a procedimiento. Esta se puede suprimir debido a que esta es implícita en la finalización de una línea de activación.

- 4) Etiquetas de mensajes: Las etiquetas de los mensajes deben tener la siguiente sintaxis:

*predecessor guard-condition sequence-expression return-value := message-name  
argument-list*

*predecessor*: Es una lista de una secuencia de números separados por coma. El significado es que no se habilita el curso de un mensaje, hasta que no se haya pasado por los mensajes de la secuencia de números listada.

*sequence-expression*: Es una secuencia de términos que terminan con ':'. Cada término representa un nivel procedural dentro de toda la acción.

*return-value*: Es una lista de nombres que designan los valores devueltos por el mensaje. Estos argumentos pueden ser utilizados como parámetros para los siguientes mensajes.

*message-name*: Es el nombre del evento que ocurre en el objeto destino (que generalmente es el evento que pide la realización de una operación).

*argument-list*: Es una lista de argumentos separados por coma, ahora parámetros, encerrados entre paréntesis. El paréntesis se puede utilizar incluso si no hay parámetros.

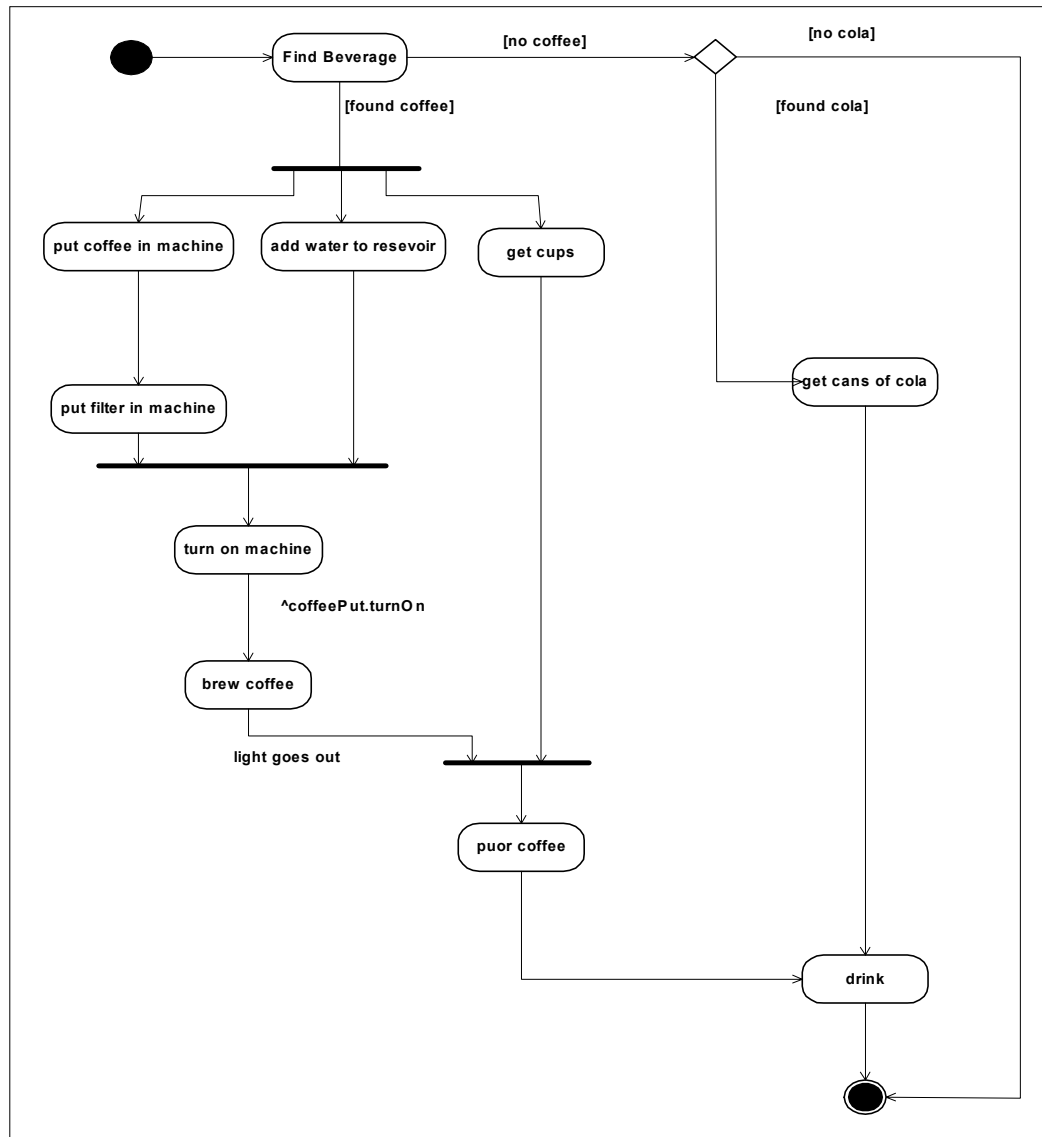
## 2.4.7 Aplicación de diagramas de colaboración a Agentes

Una de las diferencias existentes con el diagrama de secuencia es que en este diagrama los agentes pueden ser situados en cualquier lugar del gráfico, dando una mayor facilidad al momento de hacer el diagrama. En cuanto al significado del diagrama es el mismo que el diagrama de secuencia, teniendo en cuenta que la ejecución o llamadas de los mensajes están dados en las etiquetas de los mismos por el número que tienen.

Visto lo anterior quedará a parecer del que efectúe el análisis y diseño del sistema que diagrama utilizar o bien utilizar ambos.

## 2.4.8 Diagramas de Actividades

- 1) Definición: Un modelo de actividades es una variación de la máquina de estados, en la que cada estado son actividades que representan la ejecución de operaciones y las transiciones son disparadas por la ejecución completa de una operación. El propósito de este diagrama es enfocar en el flujo interno de los procesos.
- 2) Notación: Se utilizan los diagramas de actividades en los casos en los que todos o la mayoría de los eventos representan el cumplimiento de acciones generadas internamente. Para el caso de que ocurran eventos de manera asincrónica es recomendable utilizar diagrama de estados. Se muestra un ejemplo del diagrama de actividades en la figura 2.22



**Fig 2.22 Diagrama de Actividades**

- 3) Estado de acción: Una acción de estado es una forma de representar un estado con una acción interna y por lo menos una transición saliente que implica el evento implícito que completa la acción interna (pueden existir muchas de esas transiciones si conservan con algunas condiciones). La utilización normal de un estado de acción es modelar un paso en la ejecución de un algoritmo.

Un estado de acción se representa mediante un rectángulo con vértices con sus lados en forma de arcos. La expresión de la acción se escribe dentro de este símbolo. Estas expresiones deben ser únicas dentro del diagrama.

- 4) Decisiones: Una decisión puede ser mostrada mediante diferentes etiquetas a la salida de múltiples transiciones de una acción con diferentes condiciones.

El icono provisto para las decisiones es un rombo, con una o más flechas entradas y con dos o más flechas de salida, cada una de ellas etiquetada con la condición correspondiente.

- 5) Carriles de Natación: Las acciones pueden estar organizadas en carriles de natación. Estos son una especie de paquete utilizado para organizar las actividades propias de cada clase. Generalmente corresponden a unidades organizacionales en el modelo de negocio.

La manera de representar estos carriles es dividiendo el gráfico mediante líneas sólidas verticales, y cada carril generado por dos líneas pertenecerá a cada una de las clases que participen en el diagrama, conteniendo cada uno las actividades correspondientes a la clase del carril.

- 6) Aplicación de diagramas de actividades a Agentes

Los protocolos de comunicación de agentes algunas veces requieren de especificaciones con una semántica muy clara en lo que se refiere al procesamiento de hilos de ejecución. Los diagramas de actividades expresan las operaciones y los eventos que disparan a los hilos de ejecución.

Proveen una representación gráfica que hace posible la visualización de procesos de una manera simple, simplificando de esta manera el diseño y la comunicación del modelo de comportamiento. Por otra parte, se puede representar cualquier tipo de procesamiento concurrente y asíncronico. También pueden representar comunicaciones simultáneas.

## 2.4.9 Diagramas de Estados

- 1) Definición: Es un gráfico de estados y transiciones que describe la respuesta de un objeto de una determinada clase en consecuencia de la recepción de un estímulo externo. Generalmente un diagrama de estados está vinculado con una clase o bien con un método.
- 2) Notación: Representa una máquina de estados. Los estados se representan por símbolos de estado (esto es, rectángulos con vértices arqueados) y las transiciones se representan mediante flechas que conectan los símbolos de estado.
- 3) Objetos

Estado: Es una condición durante la vida de un objeto o una interacción durante la cual satisface alguna condición, ejecuta alguna acción, o espera por algún evento. Un objeto se mantiene en un estado por un período de tiempo finito.

Un estado se representa mediante un rectángulo con vértices redondeados. Puede tener uno o más compartimentos. Todos los compartimentos son opcionales. Deben contener:

- 1) Nombre del compartimento: mantiene el nombre del estado como un "String". Los estados que no poseen nombre son anónimos y son todos distintos.
  - 2) Compartimento interno de transición: mantiene una lista interna de acciones y actividades ejecutadas en respuesta a los eventos recibidos mientras el objeto se encuentra en ese estado, sin cambiar de estado.
- 4) Transiciones: Es una relación entre dos estados que indica que un objeto en el primer estado entrará en un segundo estado y ejecutará determinadas acciones específicas cuando un evento específico ocurra, si todas las condiciones son satisfechas.

Una transición se muestra como una flecha desde un estado (el estado origen) a otro estado (el estado destino) etiquetado por un String de transición.

## **Capítulo III**

# **Implementación de un Ambiente Multiagente Aplicando MaSE**

## **3. Capítulo III**

### **3.1 Introducción**

En este capítulo, aplicaremos los aspectos teóricos presentados hasta el momento en la realización de un proyecto con el kit de construcción LEGO MindStorm. Este kit permite la construcción de dispositivos a partir de un controlador programable (a partir de aquí RCX) con capacidad de procesamiento, memoria y comunicación infrarroja. La disponibilidad de motores y sensores básicos (de tacto, de luminosidad, etc.) hace posible la interacción con el ambiente.

El proyecto realizado en la Facultad de Informática de la Universidad de Morón [25], tiene como objetivo principal simular el juego conocido comúnmente como “La escondida”.

### **3.2 Realidad del juego**

La idea básica del juego es la siguiente: dados varios jugadores, se sortea o elige quién será (a partir de aquí el cazador) el que debe encontrar a los demás (a partir de aquí presas) que se esconderán en cualquier parte del espacio delimitado como escenario para el juego (por ejemplo, la cuadra del barrio). El cazador deberá comenzar el juego contando hasta determinado número sin posibilidad de observar el escenario (típicamente con los ojos vendados o de espaldas al mismo). Durante este tiempo las presas deberán esconderse.

Cuando el cazador termina la cuenta, avisa en voz alta el comienzo de la búsqueda. Mientras el buscador recorre el escenario en busca de las presas, estas pueden ir cambiando su escondite tratando de acercarse al punto de partida y de no ser descubiertos en el intento. Si el cazador descubre a alguna presa, ambos inician la carrera de retorno hacia el punto de partida. Las presas (aún no descubiertas) pueden en cualquier momento iniciar una carrera de retorno hacia el punto de partida intentando llegar antes que lo haga el cazador.

En carrera hacia el punto de partida, si llega primero el cazador, la presa quedará momentáneamente descalificada y esperando que algún otro jugador pueda salvarlo (esto solamente lo puede hacer la última presa que queda sin descubrir en caso de llegar antes que el cazador).

### **3.3 Modelo del juego (abstracción de la realidad)**

Se tratará de emular casi de forma exacta el juego en la realidad con la salvedad de que las presas no cambiarán su escondite durante el transcurso del juego pero sí intentarán volver al punto de partida luego de un tiempo si el cazador no los encuentra.

### **3.4 Modelización**

Los jugadores mencionados anteriormente estarán representados por los RCXs. Desde un punto de vista teórico, de ahora en adelante a los jugadores los denominaremos agentes.

### **3.5 Descripción de los agentes**

#### **3.5.1 Tipo de agentes**

Serán agentes basados en metas. Esto es, el agente primero debe conocer cuál es el ambiente en el que se encuentra, obteniendo lecturas de las variables que van variando a través del tiempo. Luego debe tomar la decisión adecuada en base a lo que quiere conseguir. Esto es, además de tener información sobre el estado y el ambiente debe requerir de algún tipo de información sobre cuáles son sus metas y las situaciones deseables.

#### **3.5.2 Percepciones**

Serán las que deberá obtener del ambiente en que se encuentra, como por ejemplo:

- 1) Tacto: es necesario para la navegación del agente en el ambiente, le permitirá reconocer obstáculos.
- 2) Luminosidad: puede utilizarse para seguir un plan de navegación dibujado en el ambiente. Deberá tenerse mucho cuidado con las condiciones de luminosidad en el momento de la programación del RCX, para que cuando se haga una reproducción se tengan en cuenta las mismas condiciones de iluminación del ambiente, caso contrario, se deberá calibrar los sensores.
- 3) Señales: son las señales que puede recibir del otro RCX y que a su vez forman parte de algún tipo de comunicación entre los mismos.

### 3.5.3 Acciones

Son aquellas que deberá ser capaz de ejercer el agente cuando se simule el juego, algunas de ellas son:

- 1) Avanzar: utilizando como accionadores a los motores ubicados en las salidas de los RCXs.
- 2) Girar: para permitir que el avance del agente no sea siempre en la misma dirección y también para permitir la evasión de los obstáculos que se pueden encontrar en el camino mientras navega.
- 3) Enviar mensajes: principalmente utilizado para comunicarse con los otros RCXs.
- 4) Detectar agentes: mediante el envío y recepción de mensajes el agente inducirá que encontró al otro agente.

### 3.5.4 Descripción de metas

Así como las acciones que tomará cada agente dependerá del rol que esté ocupando o desarrollando, lo mismo ocurrirá con las metas. Sólo existe una excepción y es la siguiente: todos los agentes tienen como meta principal Jugar a la escondida. Pero incluso esta meta tendrá su propia definición dependiendo del rol que se ocupe. Para ejemplificar un poco más a continuación se describirá la meta GANAR en los roles de cazador y presa:

- Para el rol cazador: significa ir descubriendo a todos los agentes que cumplen el rol de presa sin que ninguno de ellos llegue primero que él a la base.
- Para el rol presa: significa llegar a la base antes que el cazador cuando este lo descubre. Y ampliándola un poco más, si es el último agente escondido y llega antes que el agente cazador a la base, además de ganar él, los demás agentes (si alguno de ellos fue descubierto) ganan también.

### 3.5.5 Descripción del ambiente

Las siguientes son las propiedades con las que cuenta el ambiente en que se desarrollará la simulación:

- 1) Accesible: todo el aparato sensorial del agente le permite al mismo tener acceso al estado total del ambiente. El aparato sensorial detecta todos los aspectos relevantes para la elección de alguna acción que debe ejercer.
- 2) Determinista: el estado siguiente del ambiente estará determinado por el estado actual más las acciones tomadas por el agente.
- 3) Dinámico: el ambiente puede llegar a cambiar mientras está tomando alguna decisión. Esto es así porque el agente no se encuentra solo en el ambiente, sino que está acompañado por otros agentes.

Existen otras características que también nos serán de utilidad para seguir definiendo nuestro ambiente para los agentes: temporalidad, localidad, procesos y población.

### 3.5.6 Temporalidad

Básicamente esta característica nos muestra o indica si el tiempo en el ambiente se encuentra claramente dividido o no. Para nuestro caso podemos decir que existen tres divisiones temporales en el ambiente:

- 1) 1º División: abarca desde el momento en que el agente cazador (buscador) comienza a contar hasta un determinado número hasta que finaliza esta cuenta. En este intervalo de tiempo los agentes que ocupan el rol de presa deberán encontrar un lugar donde refugiarse para no ser descubiertos.
- 2) 2º División: abarca desde el momento en que el agente cazador (buscador) termina su cuenta y comienza a buscar a los agentes escondidos hasta que descubre a alguno de los agentes. Como se podrá inferir esta división de tiempo será diferente para cada agente presa (escondido). Además el agente cazador (buscador) entrará en esta división de tiempo tantas veces como agentes haya escondidos.
- 3) 3º División: abarca desde el momento en que un agente presa (escondido) es descubierto hasta el momento en que alguno de los dos agentes (buscador o escondido) llega a la base.

### 3.5.7 Localidad

Esta característica nos indica si una localidad en el ambiente puede o no ser distintiva para un agente en especial o no. ¿Pueden todos los agentes ser colocados de manera virtual o predefinida?. En una primera visión podemos decir que sí debido a que no significará lo mismo un refugio para un agente presa (escondido) que para el agente cazador (buscador). Por otro lado esta claramente definido que un agente buscador no irá a esconderse a un refugio, por lo tanto si existe una determinada determinación para las localidades de los agentes.

### 3.5.8 Procesos

Existe una manera de definir un ambiente que es la siguiente:

$$\text{Ambiente} = \langle \text{Estado}_i, \text{Proceso}_i \rangle$$

Esta expresión nos da la idea de que un ambiente puede cambiar de estado dependiendo del proceso que se esté ejecutando de forma autónoma en el mismo. El proceso si es propio del ambiente lo que indica que el proceso no está siendo invocado por una entidad externa. En términos informáticos podemos decir que el ambiente posee su propia CPU virtual.

Hasta el momento en la redacción de este informe se descarta la posibilidad de procesos propios disparados por el ambiente debido a limitaciones tecnológicas, con lo que los cambios de estado del mismo quedarán relacionados con los agentes.

### 3.5.9 Población

La población del ambiente está dada por la totalidad de las entidades que estén bajo consideración. Para nuestro proyecto la población estará constituida por todos los agentes (RCXs), por los obstáculos, los refugios y las líneas de navegación.

## 3.6 Descripción del escenario físico

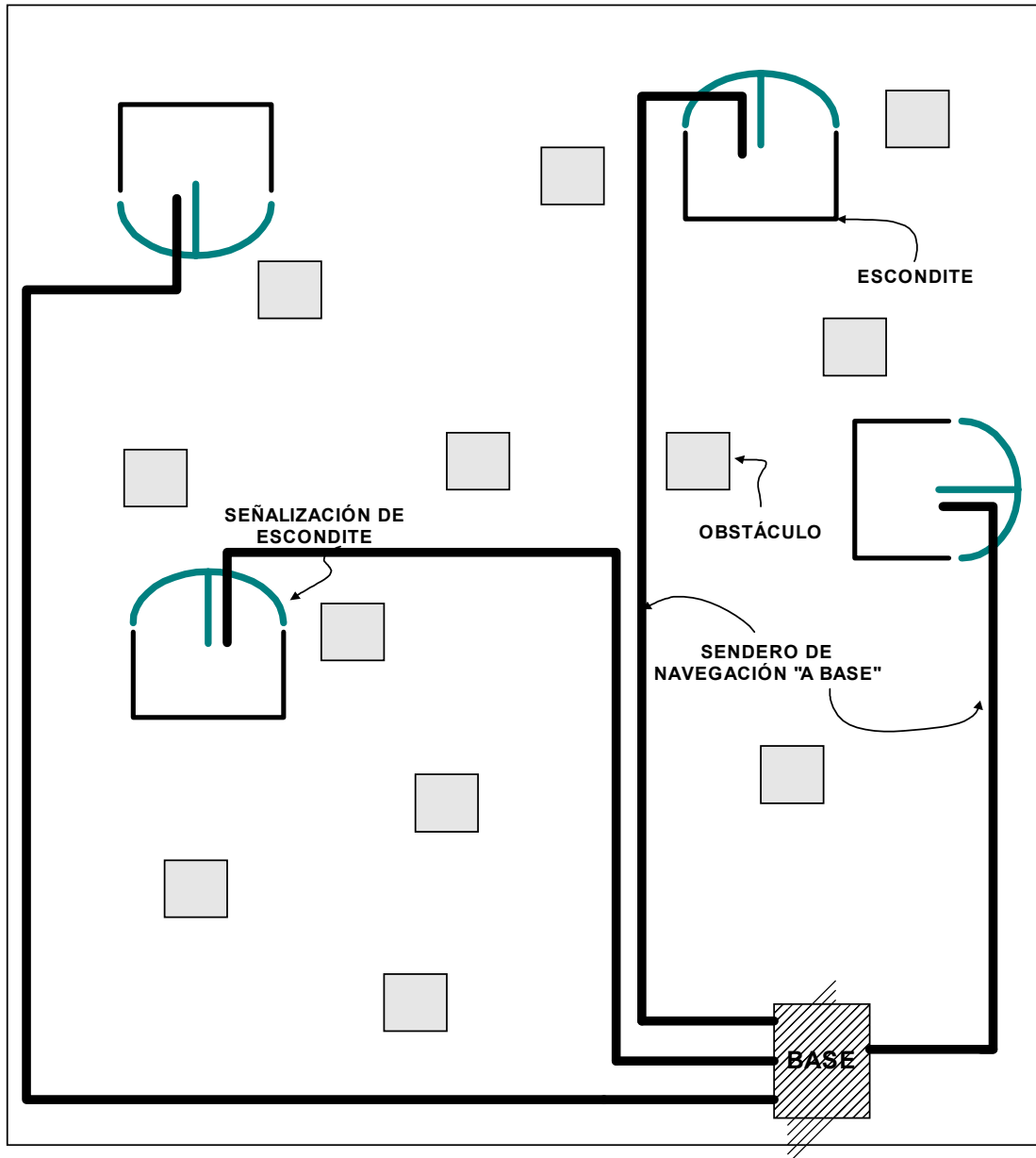
El escenario constará de una plataforma (3.20 mts. x 2.07 mts.) en la cual los agentes deberán navegar. Constará además con varios obstáculos dispersados en todo el terreno (estos no se utilizarán como lugares donde esconderse), y también de “casas” en donde los agentes podrán esconderse.

Por una cuestión de limitaciones tecnológicas, la superficie del terreno será de color blanco, esto es debido a que los sensores de luz instalados en los RCXs poseen ciertas limitaciones en cuanto a reconocimiento de niveles de luminosidad. Esto además influye en el hecho de que los niveles de

iluminación que se da al escenario en los momentos de prueba deberá ser apropiadamente documentado para luego ser reproducido de manera exacta.

La plataforma tendrá también senderos de navegación para los agentes que serán de color oscuro. Un color (marrón en nuestro caso), servirá como línea de navegación para los agentes en el momento que deban volver a la base; el otro color (negro en nuestro caso) servirá como señal para reconocer un escondite o “casa” y que el agente se esconda en él.

La imagen del ambiente o escenario en el que se desarrollo la aplicación es la siguiente:



### 3.7 Modelado utilizando la metodología MaSE

En función de la descripción previa de los agentes, a continuación presentaremos el modelo formal que se obtuvo como resultado de aplicar las fases que propone la metodología MaSE.

Basados en la documentación presente en la parte teórica de este documento, juntamente con especificaciones de UML, se utilizaron las siguientes fases y artefactos de las mismas:

- 1) Captura de metas.
  - a) Identificación de metas.
  - b) Casos de uso.
  - c) Estructuración de metas.
- 2) Transformación de metas en roles.
  - a) Identificación de tareas.
  - b) Diagrama de roles.
- 3) Diagramas de secuencia.
- 4) Ensamblado de agentes.
  - a) Construcción del diagrama de clases.

Los diagramas que se presentan han sido construidos con la herramienta AgentTool (versión 1.8.2).

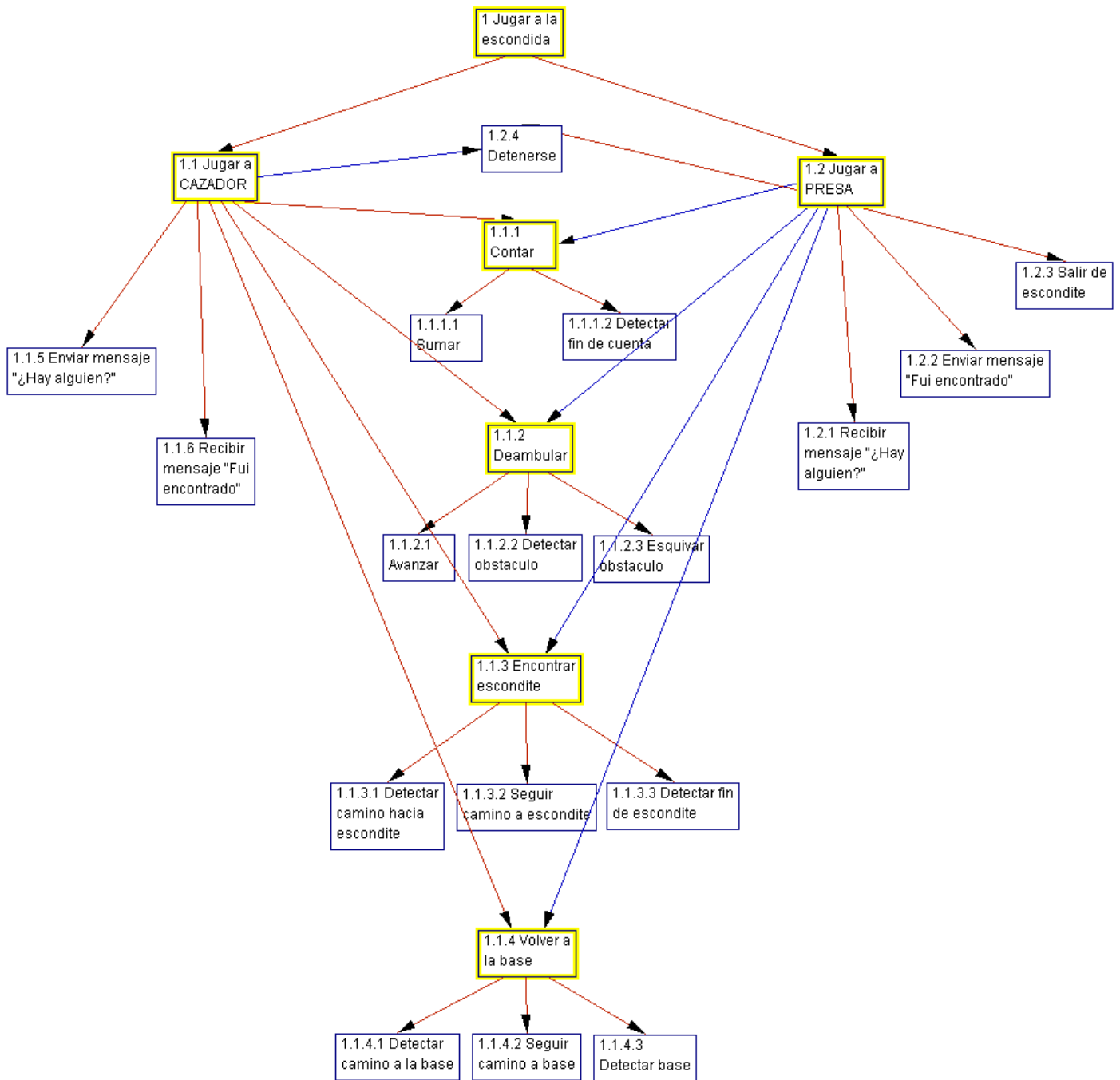
### 3.7.1 Casos de uso

- 1) Caso de uso “Jugar a CAZADOR”
  - a) Objetivo: encontrar a los agentes PRESA y retornar a la base antes que ellos.
  - b) Pre-condiciones: los agentes PRESA y el agente CAZADOR se encuentran listos para empezar a jugar. El ambiente se encuentra listo para el desarrollo del juego: iluminación requerida, obstáculos, escondites, líneas de retorno a la base, ingreso a escondites, etc.
  - c) *Trigger*: el CAZADOR comienza a *contar* y las PRESAS empiezan a *deambular*.
  - d) Post-condición: el CAZADOR a perdido (una PRESA llegó a la base antes que él) o ha ganado (el CAZADOR llegó siempre a la base antes que todas las PRESAS).
  - e) Roles intervinientes:
    - Contador.
    - Deambulador.
    - Inspector / Buscador de escondite.
    - Notificador (Cazador).
    - Buscador de base.
    - Notificador (Presa).
  - f) Descripción de los roles:
    - Comienza cuando se inicia el juego, encendiendo los RCXs. En primer lugar el CAZADOR realiza la tarea de conteo.
    - Al finalizar dicho conteo (llegando a un número prefijado), comienza con las actividades del rol de deambulador, entre las cuales se encuentran avanzar, detectar y esquivar obstáculos.
    - Al mismo tiempo, de aquí hasta que termine el juego, realizará las operaciones de notificación (rol: notificador (Cazador)), las cuales permitirán la comunicación entre CAZADOR y PRESAS, avisando a las mismas cuando fueron detectadas.
    - Al detectar un camino hacia un escondite, realiza las operaciones de esconderse, intentando detectar un agente escondido (rol: Inspector / Buscador de escondite).
    - Si en dicho escondite no había nadie, saldrá del escondite y volverá a deambular.
    - En caso de encontrar a un agente (es decir, recibe la notificación de “Fui descubierto” por una presa), procede a deambular, hasta encontrar un camino de regreso a la base, momento en que ejecutará las tareas del rol: Buscador de base, tratando de llegar antes que la presa detectada a la base (para identificar cuando llegó a la base, se colocará una base del mismo color utilizado para el camino de ingreso a los escondites).
- 2) Casos de uso “Jugar a PRESA”

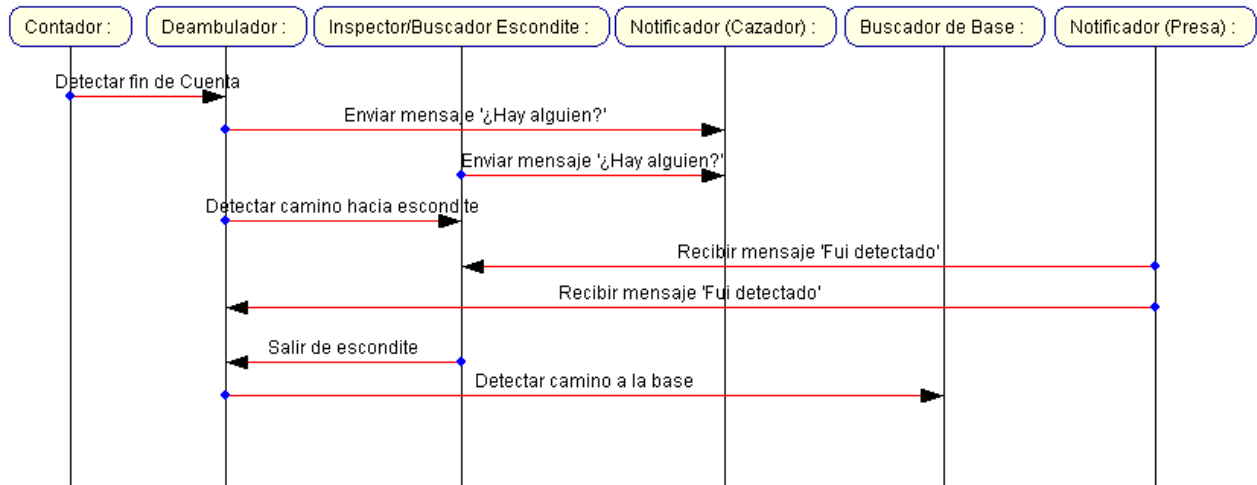
- a) Objetivo: esconderse y una vez descubierto, retornar a la base antes que el CAZADOR.
- b) Pre-condiciones: los agentes PRESA y el agente CAZADOR se encuentran listos para empezar a jugar. El ambiente se encuentra listo para el desarrollo del juego: iluminación requerida, obstáculos, escondites, líneas de retorno a la base, ingreso a escondites, etc.
- c) *Trigger*: el CAZADOR comienza a *contar* y las PRESAS empiezan a *deambular*.
- d) Post-condición: una PRESA ha ganado (es decir, llegó a la base antes que el CAZADOR) o la PRESA ha perdido llegando después que el CAZADOR a la base luego de ser descubierta.
- e) Roles intervinientes:
  - Deambulador.
  - Inspector / Buscador de escondite.
  - Notificador (Cazador).
  - Buscador de base.
  - Notificador (Presa).
- f) Descripción de los roles:
  - Comienza cuando se inicia el juego, encendiendo los RCXs. En primer lugar la presa comienza con las actividades del rol de deambulador, entre las cuales se encuentran avanzar, detectar y esquivar obstáculos.
  - Al detectar un camino hacia un escondite, realiza las operaciones de esconderse, (rol: Inspector / Buscador de escondite).
  - Al recibir el mensaje de “¿Hay alguien?” por parte del cazador, el agente enviará una confirmación de que “Fue encontrado” (rol: notificador (presa)), y en este momento saldrá del escondite.
  - Al salir del escondite, deambulará hasta encontrar un camino hacia la base (rol: deambulador).
  - Al encontrar dicho camino, ejecutará las tareas del rol: Buscador de base, siguiendo el camino hacia la misma, tratando de llegar antes que el cazador. De ser así, las presas ganarán el juego (“Piedra libre para todos”), en caso contrario, dicha presa habrá perdido, retirándose del juego.

### 3.7.2 Diagramas realizados aplicando Agent Tools

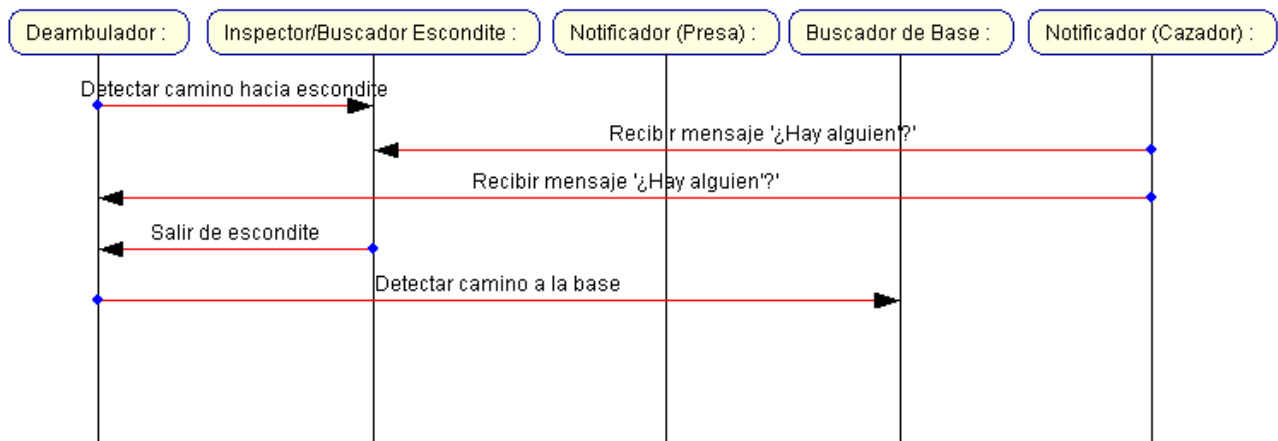
#### 3.7.2.1 Diagrama jerárquico de metas



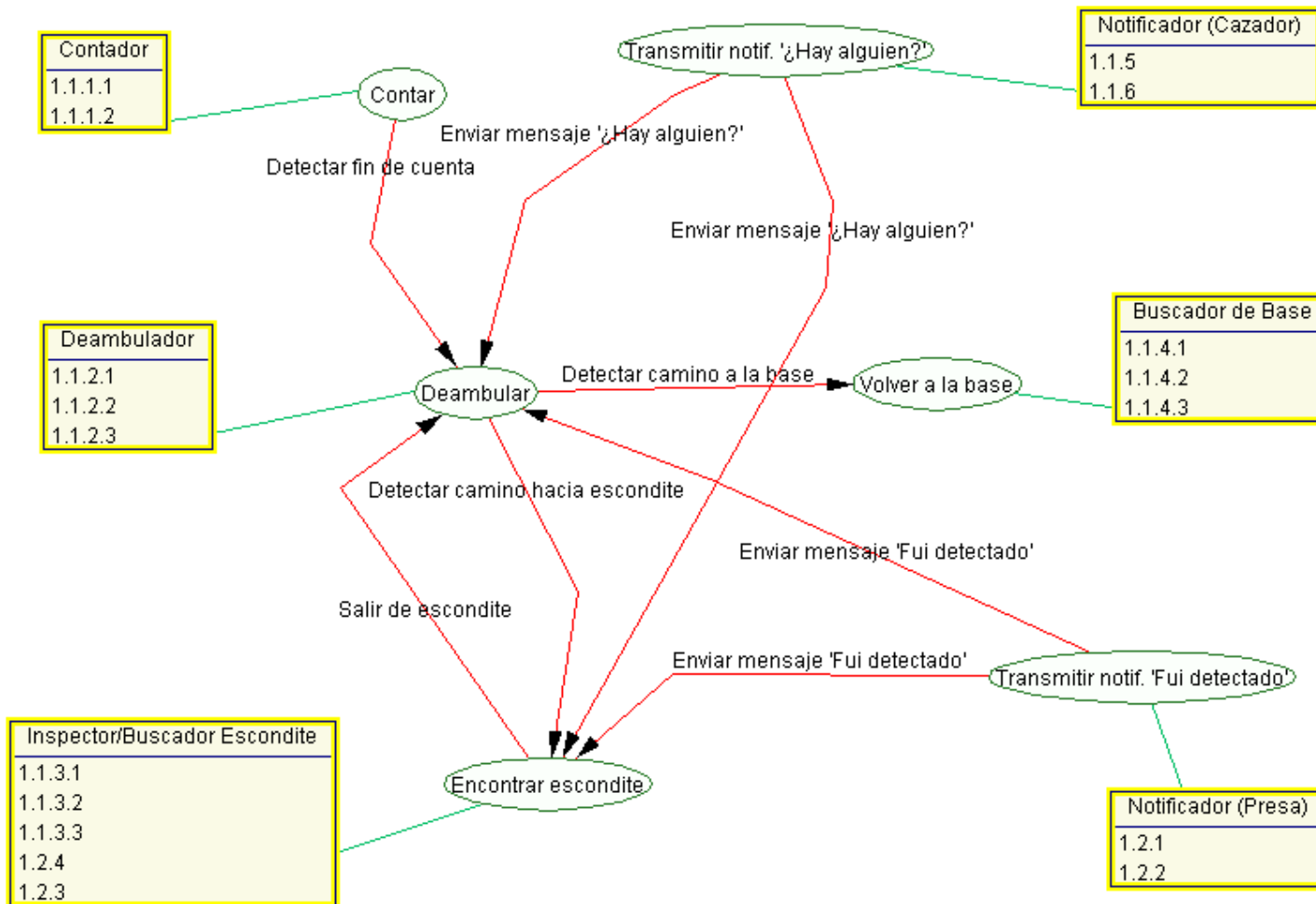
### 3.7.2.2 Diagrama de secuencia – Caso de uso “Jugar a CAZADOR”



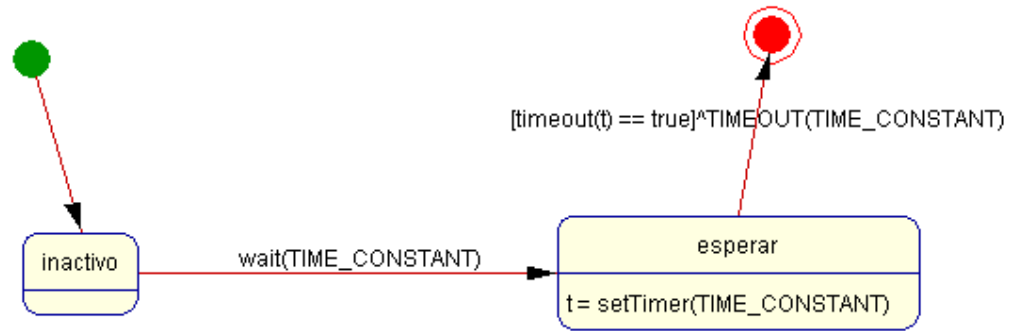
### 3.7.2.3 Diagrama de secuencia – Caso de uso “Jugar a PRESA”



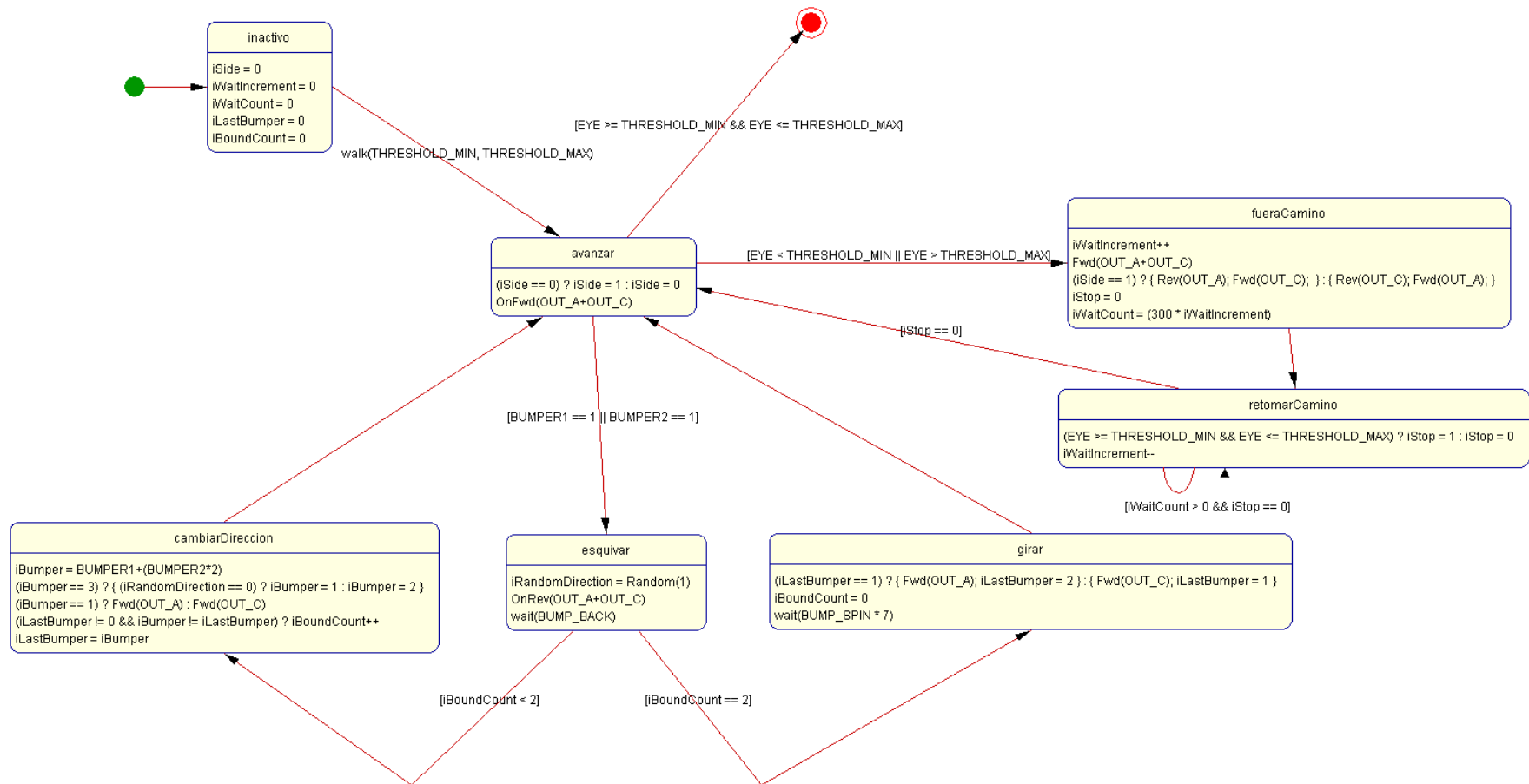
### 3.7.2.4 Diagrama de roles



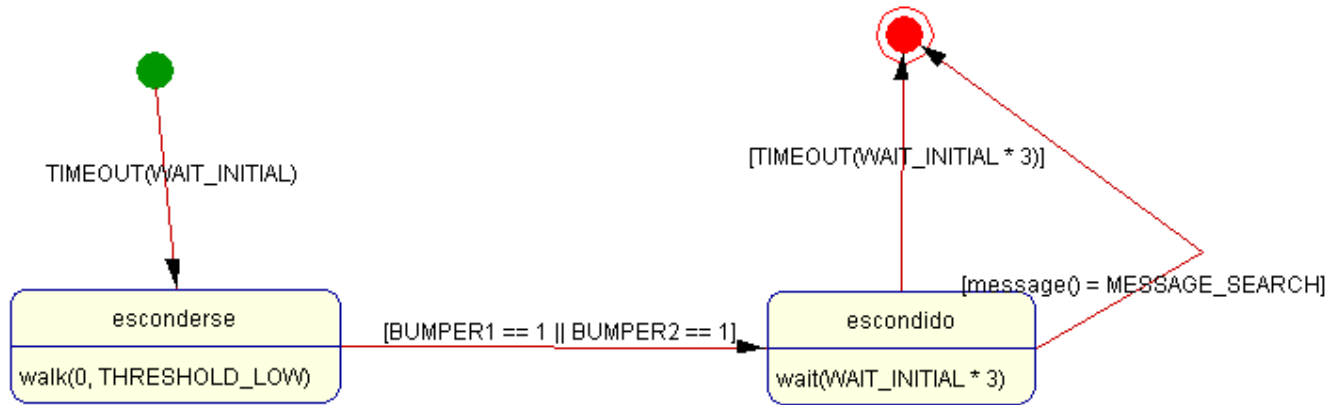
### 3.7.2.5 Diagrama de tareas – Tarea “Contar”



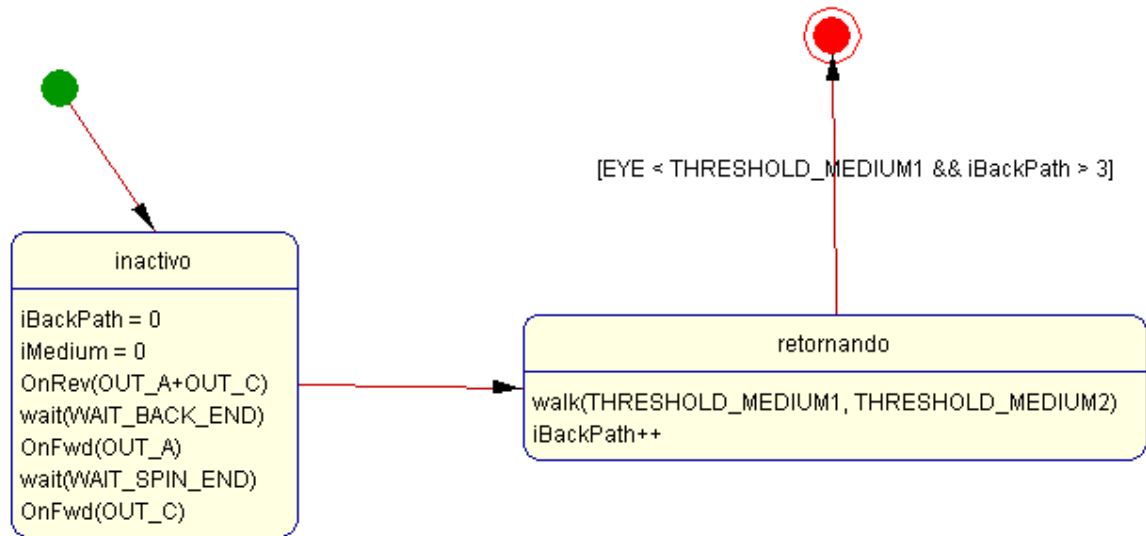
### 3.7.2.6 Diagrama de tareas – Tarea “Deambular”



### 3.7.2.7 Diagrama de tareas – Tarea “Encontrar escondite”



### 3.7.2.8 Diagrama de tareas – Tarea “Volver a la base”



### 3.8 Implementación

Para la programación de los agentes utilizamos el lenguaje NQC (Not Quite C, muy similar al lenguaje C). Los programas son incorporados a los RCXs mediante el puerto serial y la torre de transmisión que se incluye con el kit. El desarrollo de NQC fue realizado en su totalidad por [16].

La modularización de los algoritmos empleados en la automatización de los agentes fue concebida en base a los roles que estos manifiestan. Dichos roles definen los conocimientos y habilidades de los mismos para lograr una meta determinada.

La implementación descrita en esta sección se corresponde con el modelo del juego propuesto en los puntos anteriores, excepto en el caso del CAZADOR el cual “no inspecciona escondites” en busca de los agentes PRESA sino que deambula por el escenario a la espera de encontrarlos a través de los mensajes que envía.

Los roles se describen a continuación:

<b>Agente PRESA</b>		
1	Búsqueda de camino negro	El agente inicia su marcha en una dirección determinada. Al encontrar obstáculos los evita cambiando su curso. Al encontrar un camino negro cambia al rol 2. En todo momento el agente se encuentra a la espera de mensajes. En caso de recibir uno del agente buscador, este salta directamente al rol 4.
2	Seguimiento del camino negro	El agente sigue el camino, doblando al no reconocer más la presencia del color negro en el piso. Al chocar contra un objeto frena y salta al rol 3 (el agente se encuentra escondido).
3	Espera de Mensajes	El agente se encuentra frenado y a la espera de mensajes. Tiene un tiempo de espera máximo luego del cual salta al rol 4, tal como si le hubiese llegado un mensaje del agente buscador.
4	Búsqueda de meta	El agente busca la meta de llegada para poder obtener la victoria del juego.

<b>Agente CAZADOR</b>		
1	Espera de tiempo inicial	El agente espera un tiempo inicial en el cual los agentes deberán llegar a un escondite. Al cumplirse dicho tiempo salta al rol 2.
2	Búsqueda de agentes escondidos	El agente inicia su marcha en una dirección determinada. Al encontrar obstáculos los evita cambiando su curso. Constantemente envía mensajes a los agentes escondidos a la espera de una respuesta. Al recibir una respuesta salta al rol 3.
3	Búsqueda de meta	Idem rol 4 del agente PRESA.

Código NQC

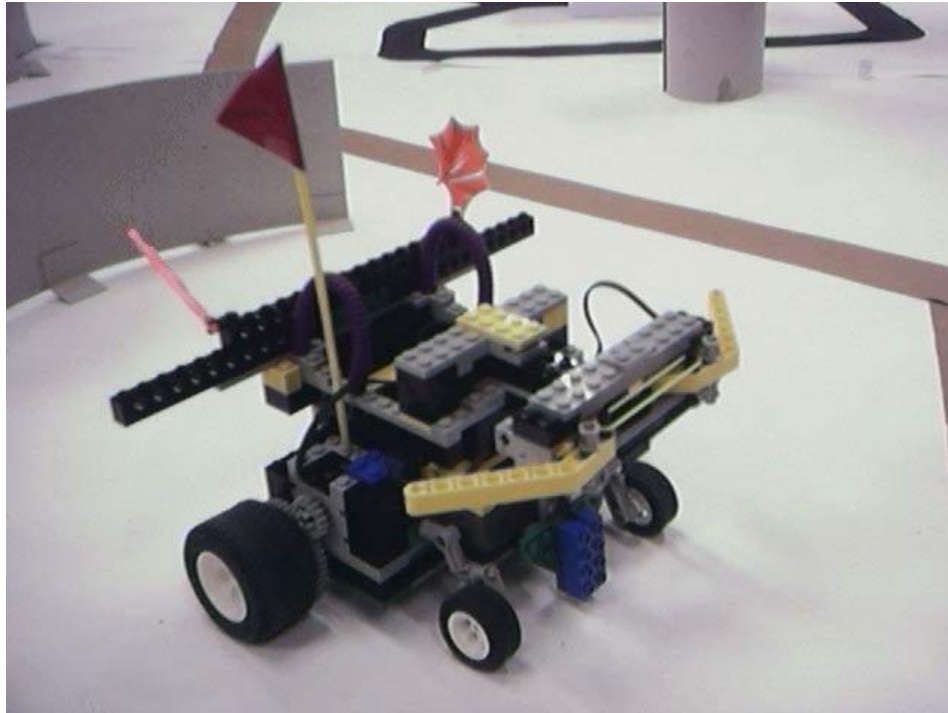
Cada uno de los roles definidos en los agentes ocupan un *slot* de programa en cada RCX. La configuración queda de la siguiente manera:

<b>Agente PRESA</b>		
1	Búsqueda de camino negro	SLOT 1
2	Seguimiento del camino negro	SLOT 2
3	Espera de Mensajes	SLOT 3
4	Búsqueda de meta	SLOT 4

<b>Agente CAZADOR</b>		
1	Espera de tiempo inicial	SLOT 1

2	Búsqueda de agentes escondidos	SLOT 2
3	Búsqueda de meta	SLOT 4

Para la programación se realizó un archivo por rol y una librería común a los mismos.



**Modelo Implementado de Agente Robot**

## **Capítulo IV**

# **Propuesta de Integración de Ingeniería del Conocimiento en Metodología Multiagente**

## 4. Capítulo IV

### 4.1 Modelización del Conocimiento en un Ambiente Multiagente

Se presenta este capítulo como principal aporte del presente trabajo, en este se propone el enriquecimiento de la metodología presentada en el capítulo anterior con la modelización del conocimiento en el desarrollo de aplicaciones multiagente considerando la adquisición, conceptualización y formalización de los conocimientos acompañado de las técnicas recomendadas para su representación.

Si bien metodológicamente se determinan las metas en un diagrama jerárquico el que identifica las mismas a nivel de que estas puedan ser tratadas por uno o mas agentes, se requiere una descripción detallada que represente el conocimiento necesario por parte de cada agente que actúa como responsable de los roles y tareas asociadas con una meta en particular.

El comportamiento de un rol se especifica mediante un conjunto de tareas, cada una demuestra un comportamiento individual que el rol ejercido por un agente debe mostrar.

El comportamiento que asume el agente en la ejecución del rol asignado requiere un conocimiento por parte del agente, la incorporación de este conocimiento en el desarrollo de software en ambientes multiagente se presenta como principal contribución del presente trabajo.

Cada rol puede ser interpretado por una clase agente, existe un mapeo con respecto a roles y clases agentes, si bien una clase agente por razones de diseño puede interpretar uno o mas roles, no obstante se considerara el conocimiento específico para cada rol en correspondencia a su clase agente.

La propuesta del presente trabajo se orienta brindar una aproximación metodológica para el desarrollo de software en ambientes multiagente para tal fin se propone considerar la metodología de base sobre la que se propone incorporar la Ingeniería de Conocimiento (Knowledge Engineering), sugiriendo como nombre MaSE & KE. Las fases y etapas propuestas pueden verse en la figura 4-1

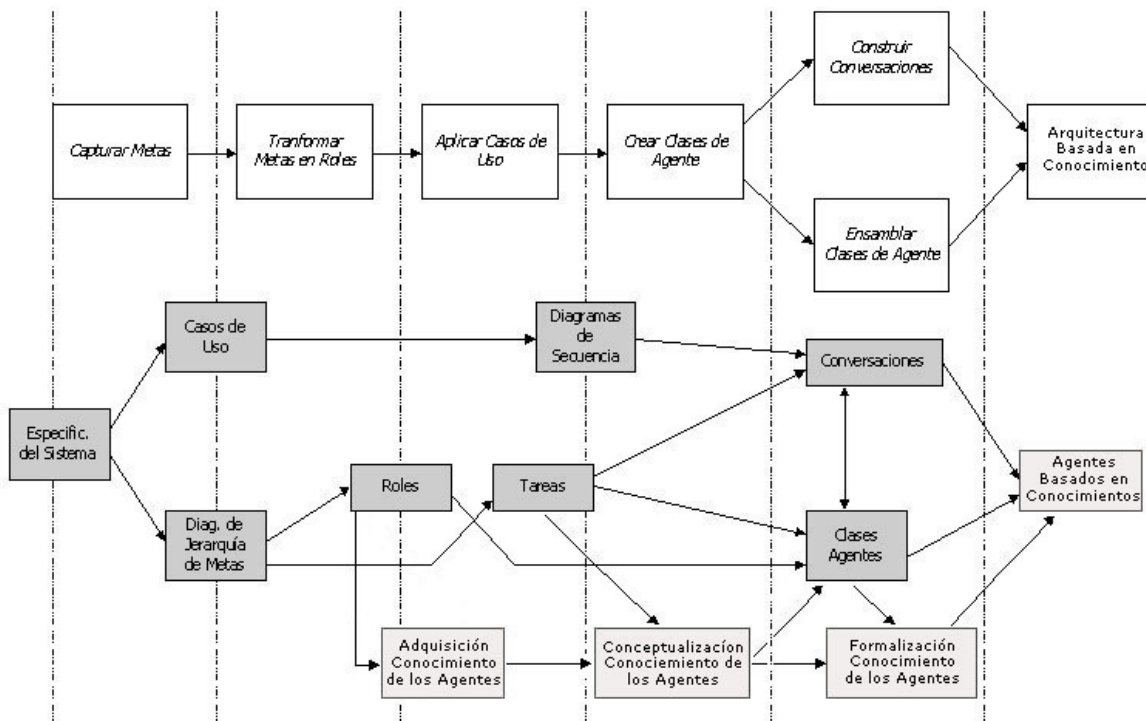


Fig. 4.1. MaSE & KE.

Para incorporar la visión de ingeniería de conocimiento se considera en primer lugar la Fase de Transformación de Metas a Roles de la metodología MaSE sobre la cual se propone incluir una etapa que contemple la adquisición de conocimiento del rol.

En segundo lugar se propone incluir en la fase correspondiente a creación de clases de agente una etapa destinada a la conceptualización del conocimiento necesario para el desempeño del rol por parte del agente.

En tercer lugar se propone incluir una etapa en paralelo con las fase de ensamblado de clases de agente y construcción de conversaciones destinada a la formalización de los conocimientos del agente cuyo objetivo es el modelado del conocimiento que el agente requiere para la ejecución de las tareas que conforman el rol.

## 4.2 Adquisición del Conocimientos de los Roles de los Agentes

La adquisición de conocimientos se extiende durante todo el ciclo de vida del agente. El mayor aporte y esfuerzo se realiza durante la etapa de la conceptualización del rol y sus tareas, lo cual se asegura a través de una adecuada adquisición de conocimientos.

La adquisición de conocimientos se presenta en dos facetas diferenciales como lo son la Extracción de conocimientos y la Educción de conocimientos. La extracción de conocimientos se relaciona con conocimientos públicos.

La educción de conocimientos identifica los conocimientos estratégicos, tácticos, fácticos y los metacimientos que cuentan los expertos para desarrollar sus tareas, las que serán realizadas por los agentes. Estos conocimientos han sido obtenidos por los expertos a través de la práctica habitual durante la resolución de su tarea, están embebidos mentalmente en el experto.

El esquema que el Ingeniero en Conocimiento debe tener del dominio en cuestión se conforma de lo general (grano grueso) a lo particular o específico (grano fino). El Ingeniero en conocimientos debe en primer lugar trabajar en los Conocimientos Estáticos del dominio, luego podrá tener las bases para pasar a las estrategias y procedimientos para la resolución del problema, de esta forma se logra un enfoque gradual para pasar de lo general al detalle determinando metas, roles y tareas de cada agente del sistema multiagente.

La educción de conocimientos tiene como propósitos determinar los conocimientos privados del experto que en la practica ejecuta el rol que llevara adelante el agente.

Las técnicas más comunes usadas para el proceso de adquisición de conocimientos del experto son para:

- 1) Primeras reuniones y evaluación de la viabilidad:
  - a) Entrevistas abiertas.
  - b) Observación de tareas habituales
  - c) Entrevistas Estructuradas.
- 2) Extracción de Conocimientos:
  - a) Análisis estructural de textos.
  - b) Clasificación de conceptos
  - c) Incidentes críticos
- 3) Educción de conocimientos del experto principal:
  - a) Análisis de protocolos
  - b) Entrevista estructuradas
  - c) Emparrillado.
  - d) Inducción

Para desarrollar la tarea de adquisición se recomienda el ciclo de educción propuesto por Gómez, A y Otros [17] para cada sesión con los expertos:

- 1) Preparación de la sesión considerando:
  - a) Información a tratar.
  - b) Amplitud, profundidad, etc.
  - c) Técnica adecuada.

- d) Preparación de las preguntas.
- 2) Sesión:
  - a) Repaso del análisis.
  - b) Explicación al experto de los objetivos.
  - c) Evaluación de la sesión con el experto.
  - d) Resumen y comentarios del experto.
- 3) Transcripción de la sesión.
- 4) Análisis de la Sesión:
  - a) Lectura para obtener una visión general.
  - b) Extracción de conocimientos concretos.
  - c) Lectura para recuperar conocimientos olvidados.
  - d) Críticas para mejorar por parte del ingeniero de conocimientos.
- 5) Evaluación:
  - a) ¿ Se han conseguido los objetivos?
  - b) ¿ Es necesario volver sobre los mismos?
  - c) Número y tipo de sesiones para cubrir la tarea

### 4.3 Conceptualización de los Agentes

De acuerdo a lo planteado por Gómez, A y otros [17], la conceptualización conlleva un proceso de estructuración de los conocimientos adquiridos que se representa en la Figura 3-2. Este se lleva a cabo en dos etapas. La primera se corresponde con una actividad de análisis donde se detectarán los Conocimientos Estratégicos, Tácticos y Fáticos (representados por líneas gruesas y por líneas finas respectivamente) dichos conocimientos son la base para los modelos de Síntesis. En la segunda etapa se obtiene el Modelo Dinámico y Estático como resultado de la síntesis, los que se integraran en el Mapa de Conocimiento, conformando el Modelo Conceptual del Sistema. Entender la conceptualización requiere entender cómo su estructura implementa una función, es decir cómo su Modelo Estático implementa su Modelo Dinámico y ambos modelizan el comportamiento del experto el que en nuestro caso empleara el agente en el desempeño de su rol

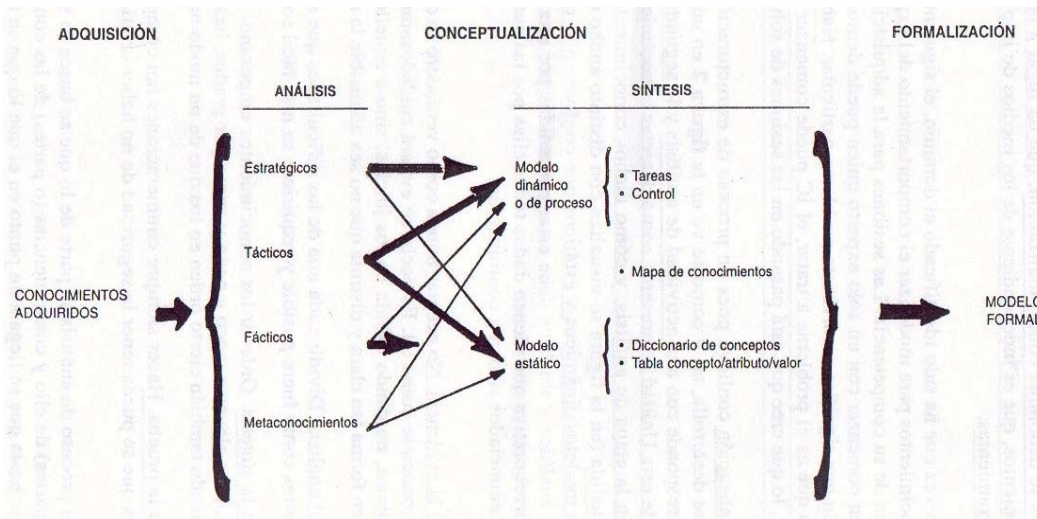


Fig. 4.2. Análisis y Síntesis para la Conceptualización [17]

#### 4.3.1 Análisis de los conocimientos

Durante la conceptualización se determinan tres tipos de conocimientos a saber, los Estratégicos que especifican qué hacer, dónde y por qué hacerlo, es decir estos conocimientos fijan la secuencia de pasos que el agente deberá seguir para ejecutar su tarea, los conocimientos Tácticos de Acción u Operativos, que especifican cómo y cuándo el agente puede añadir a sus conocimientos genéricos información actual acerca del caso y los conocimientos Fáticos o Declarativos, que especifican lo que es, o se cree que es verdad acerca del mundo en general y acerca del caso particular para el cual se está ejecutando la tarea. Finalmente también se presentan Metaconocimientos, éstos registran

la forma en que el experto usa los conocimientos para tomar una decisión, pueden existir metaconocimientos en cualquier nivel, es decir éstos pueden ser metaconocimientos del tipo estratégicos, tácticos y factuales.

### 4.3.2 Identificación, Comparación y Categorización de Conceptos

Se realiza como primer paso la identificación de los conceptos, sus atributos y valores asociados, los conocimientos fácticos especialmente se presentan a través de la realización de:

- **Glosario de términos:** Se detallan en la tabla 4-1 el modelo propuesto el que tiene como propósito documentar el significado de los términos que usa el experto en la resolución de las tareas que conformaran el rol del agente.
- **Diccionario de Conceptos:** Se identifican en la tabla 4-2 los conceptos funcionales de mayor nivel, detallando su función, sinónimos/acrónimos, los atributos que lo definen y la derivación de los datos.
- **Tabla de Conceptos - Atributos - Valores:** En la tabla 4-3 se registran los atributos propios de cada concepto que es requerido para el modelo de la tarea del experto. que el agente realizar en la ejecución de su rol.

Glosario de Términos	
Termino	Descripción

TABLA 4.1.

Diccionario de Conceptos				
Concepto	Función	Sinónimo / Acrónimo	Atributos	Derivado de

TABLA 4.2.

Concepto Atributo Valor		
Concepto	Atributo	Valor

TABLA 4.3.

### 4.3.3 Identificación de las Relaciones entre Conceptos

El segundo paso para la conceptualización consiste en identificar las relaciones entre conceptos. Se trabaja con conocimientos fácticos, se representa el modelo mental que el experto tiene del aspecto estático del problema, elaborando un modelo entidad relación. entre los conceptos identificados en la tarea del experto.

### 4.3.4 Identificación de los Conocimientos Estratégicos

El tercer paso de la conceptualización una vez identificados los conceptos, atributos y sus relaciones, comprende la identificación de las funciones del proceso de resolución del experto, los que se encuadran dentro de los conocimientos del tipo estratégico. Se representan los pasos modulares y el flujo de control de la tarea del experto a través de un gráfico de árbol de descomposición funcional.

Posteriormente se describirán los módulos de acuerdo al siguiente detalle en función de la propuesta de Gómez y otros [17]:

- **Pasos de alto nivel:** se corresponde con el primer y segundo nivel del árbol de descomposición funcional del problema.
- **Subpasos de la tarea:** se corresponde con el tercer nivel del árbol de descomposición funcional del problema.
- **Subpasos de bajo nivel:** comprende el cuarto nivel del árbol de descomposición funcional del problema.

## Descomposición Funcional de los módulos que integran el rol del agente

Para cada tarea que forma parte del rol reflejada en el árbol de descomposición funcional se detallara:

- a) Propósito de la tarea en relación al rol de ejecución del agente.
- b) Origen información representada por los atributos de Entradas del modulo o tarea en cuestión.
- c) Destino información representada por los atributos de Salida del modulo o tarea en cuestión.
- d) Razonamiento aplicado por la tarea contribuyente a su rol, detallado a través de la descripción de las acciones que conforman la tarea del experto que el agente realizara.

### 4.3.5 Identificación de los Conocimientos Tácticos

Los conocimientos tácticos del experto especifican como el agente puede usar hechos conocidos y las hipótesis acerca del caso, para obtener nuevos hechos o hipótesis tanto en situaciones deterministas como en condiciones de incertidumbre. Finalmente este análisis a producido una definición detallada de cada paso de razonamiento (identificado en el razonamiento estratégico) que deberá ejecutar el agente. La representación de Conocimiento Tácticos se efectuara a través del empleo de Seudorreglas.

### Análisis de los Conocimientos Tácticos- Seudorreglas

El conocimiento se representa contemplando el modelo de hoja de reglas según la propuesta de Gómez y otros [17], de acuerdo al siguiente modelo presentado en la tabla 4-4:

Estado de La regla	Texto de la regla
Palabra del experto	
Formulación externa de la regla	
Nombre de la regla	

TABLA 4.4.

Para el empleo de las hojas de reglas se considera:

- 1) Las palabras del Experto: En función de la fase de adquisición, identificando las reglas bajo el formato **Sí** " C1, C2,... Cn" **Entonces** " A1, A2,... An". Resultando este formato claro para el experto.
- 2) Formulación externa de la regla: Partiendo de la identificación de las reglas del experto se formalizan las reglas considerando la categorización de conceptos realizada en el primer paso del proceso de conceptualización.
- 3) Nombre de la regla: Es un nombre que identifica a cada una de las reglas empleada para la implementación. Se propone agrupar lasseudorreglas en función de la correspondencia agente-rol-tarea:

### 4.3.6 Identificación de los Conocimientos Fácticos

Durante este paso se realiza la recopilación de cada atributo de acuerdo al formato propuesto por Gómez y otros [17], según se detalla en la tabla 4-5. Este es el paso final del análisis de los conocimientos adquiridos para el desarrollo de un modelo conceptual. Los conocimientos fácticos del agente contienen información que el sistema conocerá a priori acerca del área de aplicación, así como información que el sistema obtendrá acerca del caso específico al ejecutar la tarea.

<i>Información</i>	<i>Descripción</i>
<b>Nombre</b>	
<b>Concepto</b>	
<b>Descripción</b>	
<b>Tipo valor</b>	
<b>Rango de valores</b>	
<b>Nro. Valores por caso</b>	
<b>Fuente</b>	
<b>Detalles acerca del método para obtener esta información</b>	
<b>Confiabilidad de los datos de entrada</b>	
<b>Uso</b>	
<b>Formato de los resultados de salida</b>	
<b>Material de Soporte</b>	

TABLA 4.5.

### 4.3.7 Síntesis de Conocimientos

Efectuado el proceso de análisis de los conocimientos Estratégicos, Tácticos, Fácticos, la síntesis de estos conocimientos se presenta en el Modelo Dinámico (modelo de procesos) y en el Modelo Estático, como paso final de la Conceptualización se integran ambos modelos en el Mapa de Conocimientos.

### 4.3.8 Síntesis- Modelo Estático

Este modelo está formado por los siguientes componentes documentados oportunamente y actualizados durante el proceso de conocimientos fácticos. Se detallan los componentes del Modelo Estático:

- Glosario de términos.
- Diccionario de conceptos.
- Tabla de concepto atributo valor.
- Modelo relacional de los conceptos.

### 4.3.9 Modelo Dinámico (Modelo de Procesos)

El Modelo Dinámico parte de la identificación de los conocimientos estratégicos, definiendo una jerarquía entre tareas a partir del árbol de descomposición funcional comprobando que no hay olvidos y errores. El experto en la realización del Modelo Dinámico tiene participación en la comprobación de las metas, submetas, decisiones, conceptos y atributos que se aplican en el rol del agente. El modelo se presenta con un diagrama jerárquico donde para cada modulo se detalla los atributos correspondientes a cada meta relativa a entradas y salidas producidas para cada modulo que conforma el rol del agente, ampliando a través de tablas de proceso de tareas, según se muestra en la tabla 4-6 donde se detalla el propósito de la tarea, la información necesaria para la tarea, las acciones a realizar por el agente con relación a la tarea.

<b>Identificación de la tarea</b>
<b>Propósito:</b> .
<b>Información necesaria:</b> .
<b>Acciones:</b>

TABLA 4.6. DESCRIPCIÓN DEL PROCESO DE LA TAREA

### 4.3.10 Mapa de Conocimientos

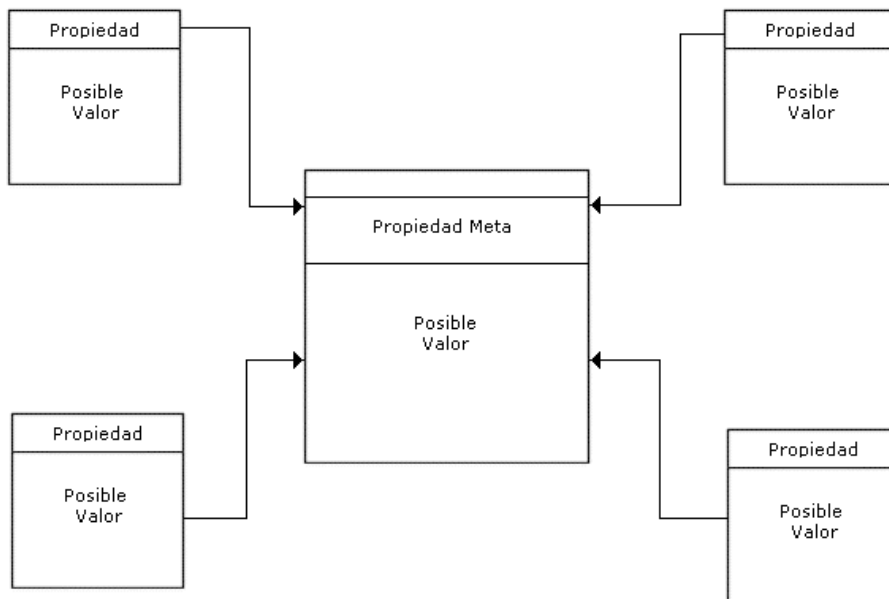
El Mapa de Conocimientos (MC) es similar a un mapa mental [Pazos j. 1996], representa el proceso de inferir valores de los atributos. Los enlaces entre los atributos y los valores inferidos forman una parte importante de los conocimientos [17].

El enfoque a través de Mapas de Conocimientos permite que los conocimientos educidos puedan ser aplicados e implementados de una forma verificable, documentable y auditable, es así que cada atributo que aparece en el lado izquierdo de una regla debe aparecer en el mapa de conocimientos conectado con el atributo que se infiere en el lado derecho de la regla, cada atributo alrededor de la decisión meta debe estar implicado en inferir o calcular el valor de la decisión meta.

El MC es la síntesis del Modelo Dinámico y del Modelo Estático, representando de esta forma la parte estática y dinámica de los conocimientos del experto que actuara en el agente. Por ultimo el MC pone en contacto directo al experto y al ingeniero en conocimiento al representar en forma entendible a los usuarios finales los conocimientos educidos. La figura 3-3 ilustra la representación de un mapa de conocimiento genérico.

Para la construcción de mapas de conocimiento se debe:

- 1) Identificar las decisiones metas que del rol del agente, relativas a un concepto en particular el cual se representa a través de un bloque que se identifica indicando rol: concepto: atributo meta, acompañado por los posibles valores que puede adoptar el atributo meta, según se muestra en la tabla 4.2.
- 2) Añadir atributos implicados en inferir o calcular la decisión meta: Colocar atributos alrededor de esa decisión meta que se usaran en los formalismos como reglas, marcos, para inferir el valor de cada decisión meta, como así también los cálculos para determinar la decisión meta.
- 3) Se engloba con línea discontinúa el conjunto de atributos meta que participan en el rol del agente, los bloques de atributos contribuyentes a los atributos metas quedan fuera del limite identificado por líneas discontinuas.
- 4) Para identificar los atributos duplicados en el mapa de conocimiento estos se marcan en el bloque el cual solo contempla su nombre con un asterisco



**Figura. 4.3. Mapas de Conocimientos (MC)**

### 4.3.11 Comprobación de la conceptualización

Las comprobaciones una vez finalizado los Modelos Estático, Dinámico y su síntesis el Mapa de Conocimientos (MC) tiene como propósito eliminar subjetividades, considerar condiciones desconocidas, confrontar la incertidumbre, verificar la completud y consistencia del modelo global.

Los pasos a realizar a fin de efectuar la comprobación de la Conceptualización, se enumeran a continuación:

- 1) Verificación para asegurar que todos los atributos de la periferia del MC son ingresados por el usuario o obtenidos a través de archivos externos.
- 2) Verificación para asegurar que todos los atributos inferidos no fueran subjetivos y que estuvieran contenidos en reglas.
- 3) Verificación para asegurar que los valores desconocidos de los atributos fueran un valor por omisión.
- 4) Verificación para asegurar que todos los atributos periféricos o inferidos se encuentren en la tabla Concepto - Atributo - Valor.

## 4.4 Formalización de Conocimientos de los Agentes

Esta fase tiene como propósito presentar el resultado obtenido de la formalización a partir de la conceptualización de conocimientos representada en arboles de descomposición funcional, seudorreglas, tablas concepto-atributo-valor determinados en la conceptualización. Una representación formal crea modelos formales que brindan una representación semi-interna o semi-computable de los conocimientos y conducta del experto que puedan ser utilizadas por el agente en la ejecución de su rol.

### 4.4.1 Selección de Formalismos

Considerando los formalismos de representación empleados en la fase de conceptualización, los formalismos seleccionados para la formalización son:

- Reglas de producción para las seudorreglas ya que su estructura es la misma.
- Marcos para la tabla concepto-atributo-valor.
- Procedimiento para los procesos a realizar.

### 4.4.2 Formalización de los Conocimientos en Reglas de Producción

Durante la fase de conceptualización correspondiente a la fase anterior se documentaron las seudorreglas, considerando la semejanza de éstas con las reglas de producción, se desarrolla las reglas de producción en un solo paso orientado al ambiente de implementación, se propone el modelado visual de reglas de la base de conocimiento del agente, partiendo de la propuesta de J Schummler [18], considerando la importancia de la actualización del conocimiento del agente es oportuno garantizar un mecanismo de representación que asegure el mantenimiento de la base de conocimiento del agente, la idea es mostrar de una manera concisa las conexiones entre reglas, así como su contenido y con ello comunicar con claridad la naturaleza de la base de conocimiento del agente.

### 4.4.3 Modelado de la base de conocimiento de los Agentes

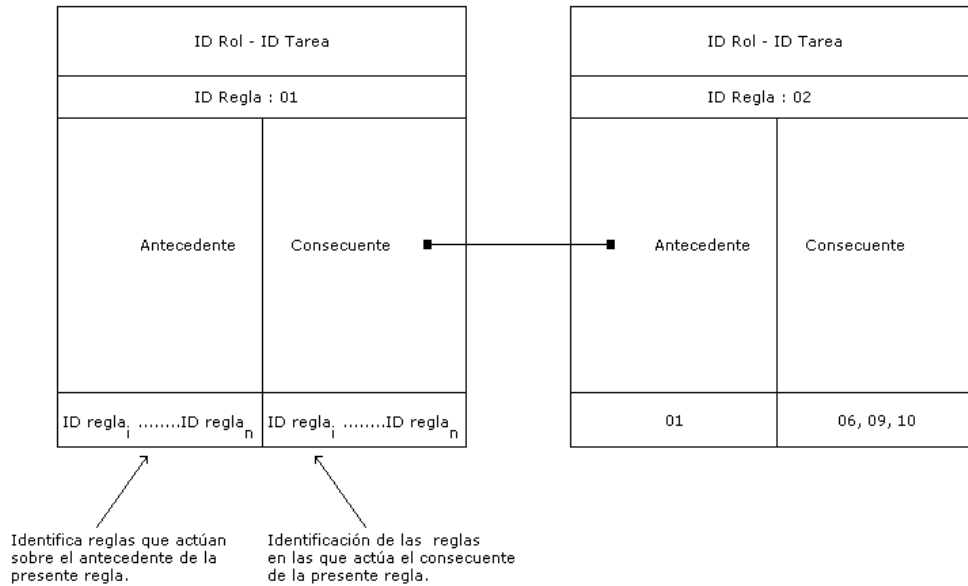
Se propone en esta fase la representación de la base de conocimiento del agente bajo el espíritu de UML, considerando la importancia de la actualización del conocimiento del agente es oportuno garantizar un mecanismo de representación que asegure el mantenimiento, la idea es mostrar de manera concisa las conexiones entre reglas, así como su contenido y con ello comunicar con claridad la naturaleza de la base de conocimiento del Agente.

El modelado visual de reglas propuesto se muestra en la figura 4-4, esta representación considera una caja por cada regla con tres compartimentos superior central e inferior.

El compartimento superior se presenta el identificador de rol, tarea y regla, en el lado izquierdo del compartimento central el antecedente de la regla correspondiente al If, en el lado derecho del compartimento central el consecuente correspondiente al Then, en el lado izquierdo del compartimento inferior identificación de reglas cuyos

consecuentes (partes Then) se relacionan con la parte de If de esta, en el lado derecho del compartimento inferior identificación de reglas cuyos antecedentes (partes If) se conectan con la parte Then de ésta.

Gráficamente la relación entre reglas se representa a través de la unión de cajas por medio de una línea entre consecuentes y antecedentes.



**Fig. 4.4. Modelado Visual de Reglas**

#### 4.4.4 Formalización de los Conocimientos en Marcos

El formalismo de Marcos es una de las técnicas más empleadas cuando el conocimiento del dominio está organizado en base a conceptos. Minsky [19] definió los Marcos como una estructura de datos que representa situaciones estereotipadas construida sobre situaciones similares ocurridas anteriormente, permitiendo así aplicar a situaciones nuevas el conocimiento de situaciones, eventos y conceptos previos.

Los Marcos agregan una tercera dimensión al permitir que los nodos tengan estructuras, que pueden ser valores simples u otros marcos [20].

A través de formalismos de Marcos se representan los conceptos y sus atributos determinados en la fase conceptualización, los conceptos de la tabla concepto- atributo- valor se formalizan en Marcos clase, los atributos del concepto representan las propiedades del Marco. Los valores de cada atributo correspondiente a las propiedades del Marco se detallan a través de las facetas que expresan de múltiples formas los valores con los que se puede rellenar cada propiedad. Las facetas, independientemente se rellenen con valores, punteros o procedimientos, se clasifican en tres categorías:

- Facetas que definen propiedades de la clase, instancia y relación, representadas por la faceta tipo ranura, cardinalidad mínima y cardinalidad máxima de valores que puede tomar la propiedad y multivaluada si la propiedad toma más de un valor.
- Facetas que define propiedades de clases y relaciones, representadas por propiedad general.
- Facetas que definen propiedades de instancia, las más típicas son valores permitidos de la propiedad, valores por omisión o por defecto asignados a la propiedad, si necesito, si añadido, si modifico, si borro, que se utilizan al necesitar añadir, modificar, borrar un valor en una propiedad.

La tabla 4.7 muestra un marco clase genérico con sus distintas facetas.

Marco Clase (MC)	Tipo Ranura	Min/Max	Mult .	Propiedad General	Valores Permitidos	Valor Omisión	Demonios
Propiedad Relación	Numérico Texto Marco	1/1 1/N	No Si	-^MC	Numérico Texto		Si Necesito Si Añado Si Modifico Si borro

**Tabla 4.7. Marco clase**

#### 4.4.5 Formalización de los Conocimientos en Procedimientos

Se describen en este punto los procedimientos que se asocian a cada propiedad del marco correspondientes con la carga, validez y robustez del sistema que intervienen en la resolución del problema.

#### 4.4.6 Guiones

Los guiones son un formalismo de representación de conocimiento que se utilizan para representar secuencias estereotipadas de sucesos, un guión es una estructura que representa una secuencias de sucesos que ocurren comúnmente, un guión es similar a un marco, mientras el marco representa conceptos, el guión representa acciones en las que intervienen dichos conceptos, ambos marcos y guiones pueden aparecer conjuntamente en la base de conocimiento del agente, el guión indicara a nuestro agente un conjunto de acciones recomendadas en función de la situación.

Un guión se conforma comúnmente con los componentes:

1. Cabecera la que considera como argumentos, nombre del guión, papeles o roles que contienen los actores involucrados en el guión, apoyo que contiene las clases involucradas en el desarrollo del guión, punto de vista desde el cual se ve la secuencia de acciones que el guión describe.
2. Condición de entrada: conjunto de situaciones que se deben cumplir en la base de conocimiento del agente.
3. Condición de salida o resultados: conjunto de acciones ciertas una vez ejecutadas los eventos descritos en el guión.
4. Escenas: cada escena es una descripción de todos los eventos que ocurren en la situación descrita por el guión.

Los guiones se integran con los formalismos de marcos y reglas, estos son útiles para representar acciones e introducir ayudas en la base de conocimiento del agente.

#### 4.4.7 Sistemas de Producción

La arquitectura de un Sistema Experto se conforma por tres componentes. Estos son: la Base de Hechos (BH), la Base de Reglas (BR) y la Estrategia de Control. La Base de Hechos y la Base de Reglas forman la Base de Conocimientos del Sistema.

#### 4.4.8 Representación de Conocimientos en el Sistema de Producción

La base de hechos del sistema está representada por un conjunto de datos y hechos que reflejan la situación que percibe el agente de su entorno de actuación Básicamente los hechos se corresponde con las situaciones que percibe el agente en función de sus sensores y la interacción con otros agentes. La Base de Hechos presenta tres estados, éstos son:

- 1) Estado inicial: Representa la situación origen del problema en el que actúa el agente

- 2) Estado Meta: Representado por la situación objetivo, se logra cuando del agente alcanza su meta.
- 3) Estado Intermedio entre el Estado Inicial y Meta: El agente se basa en la resolución de submetas antes de alcanzar la meta final.

#### **4.4.9 Estrategia de Control**

La estrategia de control es el mecanismo que examina en cada ciclo de funcionamiento los datos y hechos de la base de hecho y determina la regla que disparan, la ejecución de reglas que modifican la base de hechos.

El conocimiento que informa sobre qué caminos son los más apropiados para alcanzar rápidamente un estado objetivo, se denomina conocimiento de control de búsqueda [21].

#### **4.4.10 Representación de Conocimientos sobre la solución del problema**

Los conocimientos se representan en el agente por un conjunto de producciones o reglas que conforman la Base de Reglas. El lado izquierdo de la regla, denominado condición o antecedente representa una lista de cosas a verificar en la base de hechos y el lado derecho o consecuente representa un conjunto de acciones a realizar sobre la base de hechos, siempre que todos los antecedentes sean ciertos.

#### **4.4.11 Inferencia en el sistema de producción.**

El Motor de Inferencia (MI) o Estructura de Control (EC), examina en cada ciclo de funcionamiento la base de hechos y decide qué regla ejecutar, la EC selecciona alguna regla de la base de reglas que satisfaga la base de hechos presente, mientras que los hechos de la base de hechos no satisfagan una condición de terminación o se ejecute una regla de parada. El MI actúa en forma independiente de la tarea o dominio en el que se resuelve el problema.

#### **4.4.12 Formalización del Conocimiento Estratégico.**

Las heurísticas permiten guiar el funcionamiento del motor de inferencia para resolver el problema de una forma más eficiente posible. Las heurísticas pueden introducirse aplicando Metarreglas, éstas son reglas de producción que contienen conocimiento de uso de otras reglas del agente. Este formalismo puede utilizarse como técnica de control que permite representar conocimiento específico de un determinado dominio en el que opera el agente, al focalizar el proceso de resolución del problema en el conocimiento que es relevante en cada momento. Esta focalización es llevada a cabo al aconsejar, en la etapa de restricción de una estrategia de control para un sistema de producción, cuál es el mejor conjunto de reglas que se deben ejecutar en cada momento. Lo detallado básicamente se puede construir de tres formas.

- 1) Utilizando etiquetas para identificar un conjunto de reglas que pueden seleccionarse en una circunstancia concreta. Para que las reglas sean aplicables, dicha etiquetas deben encontrarse almacenadas en la base de hechos que explota el agente.
- 2) Particionando las reglas en conjuntos o jerarquías de reglas. Las metarreglas se emplearían para seleccionar un conjunto en particular.
- 3) Ordenando las reglas de la bases de reglas y estableciendo prioridades de ejecución.

Las metarreglas presentan un mecanismo de control, consisten en presentar conocimiento de control utilizando reglas que contienen conocimiento de como manejar adecuadamente los conocimientos del dominio para mejorar la eficacia y rendimiento del sistema [17].

La formalización de los conocimientos estratégicos (que consideran los pasos de alto nivel, los subpasos de la tarea y los subpasos de bajo nivel que el experto realiza para la resolución del problema) se realiza a través de la aplicación de un Marco Clase (Global) destinado, para soportar el conjunto de reglas filtradas que satisfacen cada una de las metas del agente, este marco clase destinado a soportar el conjunto de reglas filtradas, registra a través de la faceta (valores permitidos) el conjunto de permitido de identificadores de las reglas que permiten optimizar el espacio

de búsqueda, facilitando la selección de un conjunto de reglas a ser consideradas para la actuación del agente en función de su rol.

El nombre asignado a cada propiedad de esta clase identifica la etiqueta que representa el conjunto de reglas que pueden seleccionarse en una circunstancia concreta en relación a la resolución de una meta específica del agente. Se propone documentar la correspondencia entre las etiquetas de reglas y las metas del agente según se detalla en la tabla 4.8.

Correspondencia Reglas – Metas –Id Agente:				
Nombre Etiqueta (Propiedad)	Descripción Etiqueta	Nombre del Objetivo (Meta) Asociado	Propiedades a inferir que forman parte del objetivo	Correspondencia con el árbol de descomposición funcional

**Tabla 4.8. correspondencia entre etiquetas de reglas y metas del agente**

## 4.5 Modelado del conocimiento

En primer lugar se considera el propuesto por Wielinga et.,al 1994, este describe el modelado del conocimiento que tiene un determinado agente para la realización de sus funciones, esta modelización se realiza a nivel del conocimiento, sin hacer referencia a aspectos de implementación, se describe el modelado a partir del conocimiento de tareas, el conocimiento del dominio y del conocimiento sobre inferencias.

- El conocimiento de tareas: describe la descomposición de tareas de alto nivel en varias subtareas. El conocimiento de una tarea se divide en dos partes una que sirve para especificar el objetivo de la tarea en termino de roles de entrada y de salida, por otro lado esta el método de la tarea, que define como se llevara acabo la tarea indicando en que subtarea se descompone y en que orden deben ser procesadas (control).
- El conocimiento del dominio: especifica los hechos y asunciones que necesita el proceso de razonamiento para llevar su cometido en el domino de la aplicación.
- El conocimiento sobre inferencias: describe los procesos primitivos de razonamiento que tienen lugar en una aplicación, así como los roles de conocimiento que son utilizados por las inferencias, obviamente estos roles de conocimiento están relacionados con elementos del conocimiento del dominio.

En segundo lugar consideramos el modelo de conocimiento basado en Protege II [22] el que permite la construcción de un modelo de conocimientos distinguiendo tres elementos:

- Tareas: es un problema en el mundo real, localizado dentro de una organización o una especificación abstracta de objetivos independiente del dominio que tiene que ser alcanzados. La descripción de una tarea es una especificación abstracta del problema que debe ser resuelto por el método. Debe incluir definiciones abstractas de los datos disponibles(entradas) y de las soluciones (salidas) y de las relaciones que se deben mantener entre ellos. Aunque en la especificación de la tarea no indica como se resuelve, ésta condiciona la selección del método para su resolución.
- Métodos: Un método es un modelo independiente del dominio, que indica como se resuelve el problema especificado por la tarea. Además de la especificación de la entrada y la salida de la tarea, el método posee una definición abstracta de los roles que juega el conocimiento del dominio en la resolución del problema. Los métodos pueden delegar problemas a subtareas que, a su vez pueden ser resueltas por otros métodos, formando un árbol de tareas –métodos – subtareas.
- Mecanismos: Los mecanismos se pueden considerar como métodos primitivos que no pueden ser descompuestos en otras subtareas (no admiten descomposición). La descomposición termina cuando se alcanzan los mecanismos.

Esta estrategia de descomposición ayuda a la reutilización de componentes de dos formas: por un lado los mecanismos al desarrollar funciones muy definidas pueden ser utilizados en varios dominios y por otro lado la existencia de métodos descomponibles hace más flexible la reutilización ya que permite la utilización de métodos y mecanismos alternativos para la resolución de sus subtareas.

Además de las tareas y métodos se distinguen las siguiente ontologías:

- La ontología del dominio: Incluye una descripción declarativa del dominio y representa la conceptualización del dominio de aplicación.
- La ontología de los métodos: define los conceptos y relaciones que utilizan los métodos y sirve de marco para especificar entradas y salidas, para que los métodos de resolución de problemas puedan ser utilizadas por diferentes dominios deben ser definidos en términos independientes del dominio.
- Ontología de aplicación: Sirve para extender la ontología del dominio con conceptos y relaciones propios de la ontología de métodos.

La construcción de un modelo de conocimiento se puede hacer en tres pasos:

- Formalización de la Ontología del método, incluyendo el conjunto de roles de conocimiento que utiliza.
- Definición de la Ontología de aplicación independientemente de la del método y reutilizado alguna ontología del dominio previamente definida.
- Formulación de las relaciones de correspondencia entre las ontologías de aplicación y método.

En tercer lugar se presenta la caracterización de un agente como un sistema basado en conocimientos partiendo del Modelo de experiencia propuesto en la metodologías Common Kads y Mas-Common-Kads el que describe el conocimiento necesitado por los agentes para alcanzar sus objetivos. El modelo de experiencia junto con el modelo de cooperación se denominan modelo conceptual.

La estructuración del conocimiento se define a través de un modelo de cuatro capas para modelar la experiencia:

- Nivel de dominio: define el conocimiento estático que describe los conceptos, relaciones y estructuras.
- Nivel de Inferencia: define el conocimiento de las diferentes inferencias elementales que pueden aplicarse al conocimiento del dominio.
- Nivel de Tarea: describe el conocimiento de control que determina como se combinan las inferencias elementales para conseguir un objetivo
- Nivel Estratégico contiene el conocimiento de cómo combinar tareas y tomar acciones.

De las cuatro capas solo el nivel de dominio depende del dominio en cuestión, el resto son reutilizables de otros dominios.

La estructuración de conocimiento se denomina a menudo tipado del conocimiento, se pueden distinguir los siguientes tipos de conocimientos en la mayoría de las metodologías:

Tarea: es una descripción de un objetivo o subobjetivo

- Un método de resolución de problema(PSM o Método) es una forma de realizar una tarea.
- Inferencia (mecanismo o fuente de conocimiento): es una acción (o tarea) que no es descomponible en subtareas, pero aplica conocimiento del dominio para obtener nueva información. Representa las funciones básicas o primitivas del razonamiento.
- Léxico(u ontología del modelo): describe la terminología de un dominio dado.

- Ontología (o esquema del modelo): representa las estructuras genéricas y relaciones de un dominio, es decir, dota de estructura a las entidades del léxico y describes sus relaciones.
- Modelo del dominio (o conocimiento del dominio o de la aplicación): describe una base de conocimiento con casos concretos de los términos definidos en la ontología.

#### 4.5.1 Capturando y estructurando Objetivos

El modelo propuesto por Kendall, et al [Capturing and Structuring Goals: Anlysis Patterns, 1999] resulta de utilidad en el análisis de requerimientos de software, este presenta cuatro modelos que actúan bajo un modelo de cascada entre si, partiendo del modelo de captura de Metas, seguido por el modelo de casos de metas, continuando con el modelo estructurado de casos de metas finalizando con el modelo de objetos del casos de metas, se detalla cada paso a continuación:

- La captura de Metas: comienza por al extracción de escenarios de los requerimientos de especificación, antecedentes de usuarios documentación, normas. Cada meta se determina con la pregunta de cual es el objetivo de cada escenario, es decir las metas son identificadas en la determinación del propósito de cada escenario.
- Metas y objetivos para cada escenario tienen que ser capturados e indicados explícitamente.
- Determinando el Casos Meta: cada caso de uso obtenido del escenario debe ser relacionado con una meta en particular, el objetivo de esta propuesta es determinar un caso de uso y su meta relacionada a fin de obtener el caso de uso meta(  $Meta + Caso\ de\ Uso = Caso\ Meta$ ). Si una meta esta relacionada con múltiples escenarios, cada escenario en particular y su meta representan distintas casos meta.
- Estructurando el Caso Meta: Al presentarse los casos surge que existen diferentes escenarios, metas y distintas jerarquías como así también relaciones entre estos, para capturar estas distinciones y preservar las relaciones se realiza el modelo estructurado de casos metas obteniéndose una estructura jerárquica de los casos meta.
- Esta estructuración se realiza en orden a los documentos de especificación de requerimientos y su documentación (normas, procedimientos, etc.), distinguiendo las subordinaciones y extensiones, a fin de obtener una forma jerárquica que permite una clarificación de los casos metas y sus relaciones, representado a través de una estructura de árbol .
- Casos Metas como Objetos: Este modelo tiene como objetivo evitar la redundancia y duplicación en el modelo de análisis de los Casos Meta. Una meta similar puede aparecer en mas de un caso meta, la solución se presenta al relacionar cada caso meta con un objeto específico, idénticas metas pueden ser vistas como instancias de una misma clase, el resultado final es obtener un diagrama de jerárquico de caso meta evitando redundancias.

#### 4.5.2 Determinando Roles

Las primeras fases responden a la captura de un conjunto de objetivos estructurados a partir de las especificaciones; la transformación de objetivos estructurados jerárquicamente en roles es una forma más útil para construir sistemas multiagentes.

Los roles son los bloques de construcción utilizados para definir las clases del agente y capturar los objetivos del sistema durante la fase de diseño, a fin de garantizar que los objetivos del sistema son considerados para asegurar que cada objetivo esta asociado con un rol y que cada rol. es jugado por una clase de agente.

Un rol es una descripción abstracta de una función esperada de una entidad y encapsula los objetivos del sistema al que se le ha asignado la responsabilidad de cumplimentar.

Los roles son creados para hacer algo, son similares a la noción de un actor en una obra o una oficina dentro de una empresa.

La transformación de objetivos a roles es uno a uno cada objetivo se corresponde con un rol, sin embargo hay muchas situaciones excepcionales en las que resulta útil combinar objetivos.

Objetivos similares pueden combinarse en roles únicos por conveniencia o eficiencia, los objetivos que comparten un alto grado de cohesión pueden combinarse en un solo rol. Algunos objetivos implican roles distribuidos cualquier mención de máquinas separadas u otra distribución requiere un rol para cada lado de la relación distribuida.

El modelo de rol tradicional contempla la definición del rol a alto nivel, y los roles que lo componen, donde sus nombres se identifican con los objetivos del sistema y sus líneas de comunicación entre roles derivan de los diagramas de secuencia. Cada rol identifica al objetivo u objetivos determinados en la estructura jerárquica de objetivos involucrada.

Las tareas que componen cada rol en particular se representadas generalmente por óvalos, la comunicación entre tareas concurrentes se representa con líneas punteadas. Una tarea es un conjunto de actividades y comunicaciones estructuradas gráficamente como un diagrama de estados.

### 4.5.3 Modelado de roles para agentes

El modelado de roles para sistemas orientados a agentes ofrece una muy buena aproximación para el análisis y diseño de agentes: [23] ya que considera:

- La Proactividad: el rol en un modelo de trabajo de roles juntos acompañando a un objetivo.
- La Unificación del modelo agentes, objetos y personas pueden jugar roles.
- El Particionado: El comportamiento complejo de agentes puede ser particionado en roles.
- El Diseño: la sinergia o síntesis del modelo de rol puede ser evaluada por el diseño del agente.
- El Rol Dinámico: la organización de agentes puede tomar varias formas, son dinámicas.
- La Documentación: el modelado de rol provee documentación para el marco de trabajo de agentes del sistema la cual es independiente de la implementación, ayudando a identificar que aspectos pueden ser reorganizados.
- Para el rol del agente se consideran las siguientes dimensiones o facetas:
  - Modelo del rol: descripción, contexto.
  - Responsabilidad: servicios, tareas.
  - Colaboración: interacción del rol.
  - Interfaces externas: acceso a servicios.
  - Relación con otros roles: agregación, generalización, refinamiento, secuencia de roles.

Los agentes son extensiones de objetos [Kendall E, Role Modelling For Agent System Analysis, Design, and Implementation, 2000], ellos abarcan todas las características que tiene los objetos sumando proactividad, social, reactividad y comportamiento inteligente. Las facetas consideradas para objetos no son suficientes para agentes, estos requieren nuevas facetas que abarcan mejoras como:

- Modelo del rol: descripción, contexto.
- Responsabilidad: servicios, tareas, objetivos, obligaciones, prohibiciones.
- Colaboración: interacción del rol.
- Interfaces externas: acceso a servicios.

- Relación con otros roles: agregación, generalización, refinamiento, secuencia de roles.
- Experticia: ontologías, inferencias, resolución de problemas de conocimientos.
- Coordinación y negociación: Protocolo, resolución de conflictos, permisos, conocimiento del porque otros roles están relacionados.
- Otros: recursos, aprendizaje/adaptabilidad.

Las clases no capturan eficientemente la colaboración e iteración, la colaboración es una crucial faceta en el comportamiento de un agente, la cual es considerada en el modelado del rol.

#### **4.5.4 Clases de agentes**

Las clases de agentes surgen de dos componentes :roles y conversaciones, el producto es un diagrama de clases , representados por rectángulos que identifican a la clase y su roles asignados, las conversaciones se indican con líneas de flechas que indican el iniciador y el respondedor de la conversación entre clases, con el nombre de la conversación por encima o al lado de la línea de flecha.

Hay una correspondencia entre clases y roles al igual que antes entre roles y objetivos.

La diferencia entre un diagrama de clase clásico y un diagrama de clase de agentes es que esto representan clases y conversaciones entre las clases, dado que los agentes heredan vías de comunicación entre roles cualquier vía de comunicación entre dos roles se convierte en una conversación entre sus respectivas clases.

Una conversación define un protocolo de coordinación entre dos agentes

#### **4.5.5 Ensamblado de Clases de Agentes**

El Ensamblado de clases de agentes [24], se considera a partir de una cinco modelos de estilo arquitectónico diferentes: BDI (Belief, Desire, Intention; creencias, deseos e intenciones), Reactivo, Planning (planificación), Basado en el Conocimiento, Arquitectura definida por el usuario. Cada modelo de arquitectura cuanta con un conjunto de componentes específicos, por ejemplo la arquitectura reactiva cuanta con componentes para controlador de interfaces, bases de reglas, controlador de efectores, etc.

#### **4.5.6 Diseño del sistema multiagente**

Se recurre a un diagrama de despliegue (Deployment) para mostrar los agentes y su ubicación en el sistema, la instanciación de agentes se realiza a partir de su clases al igual que la instanciación de objetos a partir de las clase de objetos. Los agentes se representan por cajas tridimensionales, las conversaciones entre agentes se representan a través de líneas entre cajas de agentes .los agentes son nombrados después de su clase ( clase-agente:: nombre-agente), la ubicación física en función de la plataforma física para aquellos agentes que se alojan en la misma plataforma se identifican con una línea punteada que los envuelve., también se puede incluir la dirección o host name.

# **Capítulo V**

## **Conclusiones y Futuras Líneas de Investigación**

## **5. Capítulo V**

### **5.1 Conclusiones del Trabajo**

La realización del trabajo de investigación permitió en primer lugar abordar los conceptos generales de sistemas basados en agentes y las metodológicas aplicadas, con especial interés en la metodología MaSE. (Multiagent Systems Engineering) del AFIT (Air Force Institute of Technology) Agent Lab

En segundo lugar el trabajo de investigación durante su desarrollo permitió generar experiencia en el empleo de la metodología multiagente MaSE con la particularidad que se realizó una implementación específica en un ambiente de agentes robóticos. que se desarrollaron empleando tres plataformas robot montadas sobre RCX Mindstorms Robotic Invention Systems,

En tercer lugar permitió elaborar una propuesta inicial sobre la base de la Metodología aplicada, para incorporar en ésta, ingeniería de Conocimientos considerando las fases de adquisición , conceptualización y formalización de los conocimiento de los agentes en función de sus roles considerando sus conocimientos estratégicos tácticos y fácticos

### **5.2 Futuras líneas de investigación**

De la experiencia adquirida del presente trabajo surgen como propuestas:

Realizar la experimentación con las extensión de ingeniería de conocimiento sobre la metodología MaSE aplicado al dominio de la robótica

Desarrollar el prototipo de una herramienta para la modelización del conocimiento en ambientes multiagente aplicados a la robótica sobre la base de los resultados obtenidos en la experimentación propuesta.

# **Capítulo VI**

## **Bibliografía**

## 6. Bibliografía

- [1] Baum, D. and Gasperi, M., Hemprl, R., Villa, L. 2000. *Extreme Mindstorms: An Advanced Guide to Lego MindStorms* A Presss
- [2] Wooldridge, M. and Jennings, N.R., 1995. *Agent Theories, Architectures and Languages: a Survey* in Wooldridge, M. and Jennings, Eds., *Intelligence Agents*, Berlin: Springer-Verlag, Vol 1, Nro 22
- [3] Kinny, D., Georgeff, M., and Rao, A., 1996. *A methodology and modelling technique for system of BDI agents* in W. Van der Velde and J. Perram, editors, *Agents Breaking Away: Proceedings of the MAAMAW 96*. Springer-Verlag: Heidelberg, Germany.
- [4] Lange, D. 1998. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley.
- [5] Kendall, E., A. 1998. *Capturing and Structuring Goals: Analysis Patterns*.
- [6] DeLoach, S., Wood, M. June 2000. *Multiagent Systems Engineering: The Analysis Phase*.
- [7] Wooldridge, M., Jennings, N., & Kinny, D. 2000. *The Gaia Methodology for Agent-Oriented Analysis and Design*.
- [8] Depke, R., Heckel, R., & Küster, J. 2001. *Improving the Agent-Oriented Modeling Process by Roles*.
- [9] Pressman, R. 1992. *Software Engineering: A Practitioners Approach*, 3rd ed. McGraw-Hill
- [10] Wood, M., & DeLoach, s. January 2001. *An Overview of the Multiagent Systems Engineering Methodology*.
- [11] Wood, M. March 2000. *Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems*. Thesis.
- [12] Robinson, D. March 2000. *A COMPONENT BASED APPROACH TO AGENT SPECIFICATION*. Thesis.
- [13] Giret, A., Cernuzzi, L., Pastor, O., & Insfrán, E. *Orientación a Objetos y Orientación a Agentes: Una Propuesta de Unificación*.
- [14] Cernuzzi, L., & Giret, A. *Methodological Aspect in the Desing of a Multi-Agent System*
- [15] Iglesias, C.A.M., Gonzalez, J.C. and Velasco, J.R. 1997. *Analysis and design of multiagent system using MAS-CommonKADS*. In AAAI 97 Workshop on Agent Theories, Architectures and Languages, Providence, RI, ATAL.
- [16] Dave Baum (<http://www.enteract.com/~dbaum/nqc>).
- [17] Gomez, A., Juristo, N., Montes, C., Pasos, J. 1997. *Ingeniería del Conocimiento*. Rd. Centro de Estudios Ramón Areces. Madrid
- [18] Schmuller, J. 2000. *Aprendiendo UML, Modelado de Bases de Conocimientos*. Pearson Educación . pag 349, 352
- [19] Minsky, M. 1975. *A framework for representing Knowledge*. MC Hill. New York.
- [20] Giarrataro, J., Riley, G. 2001. *Sistemas Expertos. Principios y Programación*. International Thompson Editores.
- [21] Rich, E., Knight, K. 1994. *Inteligencia Artificial*. MC Hill. New York.
- [22] Puerta et al. ,1992, Erikson et al., 1995, Tu et al.,1995

- [23] Kendall E, 2000. *Role Modelling For Agent System Analysis, Design, and Implementation*
- [24] Robinson DJ. 2000. *A component bases approach to agent specification*. AFIT/ENG/00M.22.
- [25] Ramirez, S, Pinto M, Wilke S, Widelec P, Salazar M, Deluca C, Caseres F, Dignazi D, Ierache, J. 2002. Cátedra de Robótica. Facultad de Informática. Univ. De Morón.

