

# **ESCUELA POLITÉCNICA DEL EJÉRCITO**

## **SEDE LATACUNGA**



### **CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

**“DESARROLLO DE UNA AGENDA MULTIAGENTE  
ORIENTADA A LA WEB PARA LA GESTIÓN DE REUNIONES  
UTILIZANDO LA PLATAFORMA JADE – JAVA EN LA  
ESCUELA POLITÉCNICA DEL EJÉRCITO SEDE  
LATACUNGA”**

**PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE  
SISTEMAS E INFORMÁTICA**

**EITHEL JENNIFER NAVARRETE CARREÑO**

**Latacunga, Agosto 2008**

**ESCUELA POLITÉCNICA DEL EJÉRCITO**  
**CARRERA DE INGENIERÍA DE SISTEMAS E**  
**INFORMÁTICA**

**CERTIFICADO**

ING. JOSÉ LUIS CARRILLO (DIRECTOR)  
ING. EDISON ESPINOSA (CODIRECTOR)

**CERTIFICO:**

Que el trabajo titulado “DESARROLLO DE UNA AGENDA MULTIAGENTE ORIENTADO A LA WEB PARA LA GESTIÓN DE REUNIONES UTILIZANDO LA PLATAFORMA JADE – JAVA EN LA ESCUELA POLITÉCNICA DEL EJÉRCITO SEDE LATACUNGA” realizado por la señorita EITHEL JENNIFER NAVARRETE CARREÑO ha sido guiado y revisado periódicamente y cumple normas estatutarias establecidas por la ESPE, en el Reglamento de Estudiantes de la Escuela Politécnica del Ejército.

Debido a que constituye un trabajo de excelente contenido científico que coadyuvará a la aplicación de conocimientos y al desarrollo profesional, SI recomiendan su publicación.

El mencionado trabajo consta de un empastado y un disco compacto el cual contiene los archivos en formato digital. Autorizan a la señorita EITHEL JENNIFER NAVARRETE CARREÑO que lo entreguen al ING. EDISON ESPINOSA, en su calidad de Coordinador de Carrera.

Latacunga, 14 de Agosto del 2008

---

Ing. José Luis Carrillo  
**DIRECTOR**

---

Ing. Edison Espinosa  
**CODIRECTOR**

**ESCUELA POLITÉCNICA DEL EJÉRCITO**  
**CARRERA DE INGENIERÍA DE SISTEMAS E**  
**INFORMÁTICA**

**AUTORIZACIÓN**

Yo, EITHEL JENNIFER NAVARRETE CARREÑO

Autorizo a la Escuela Politécnica del Ejército la publicación, en la biblioteca virtual de la Institución, el trabajo de grado titulado “DESARROLLO DE UNA AGENDA MULTIAGENTE ORIENTADO A LA WEB PARA LA GESTIÓN DE REUNIONES UTILIZANDO LA PLATAFORMA JADE – JAVA EN LA ESCUELA POLITÉCNICA DEL EJÉRCITO SEDE LATACUNGA” cuyo contenido, ideas y criterios es de mi exclusiva responsabilidad y autoría.

Latacunga, 14 de Agosto del 2008

---

Eithel Jennifer Navarrete Carreño

C.I. No. 091994833-1

**ESCUELA POLITÉCNICA DEL EJÉRCITO**  
**CARRERA DE INGENIERÍA DE SISTEMAS E**  
**INFORMÁTICA**

**DECLARACIÓN DE RESPONSABILIDAD**

Yo, EITHEL JENNIFER NAVARRETE CARREÑO

**DECLARO QUE:**

El proyecto de grado denominado “DESARROLLO DE UNA AGENDA MULTIAGENTE ORIENTADO A LA WEB PARA LA GESTIÓN DE REUNIONES UTILIZANDO LA PLATAFORMA JADE – JAVA EN LA ESCUELA POLITÉCNICA DEL EJÉRCITO SEDE LATACUNGA” ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en las referencias bibliográficas.

Consecuentemente este trabajo es de mi autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Latacunga, 14 de Agosto del 2008

---

Eithel Jennifer Navarrete Carreño

C.I. No. 091994833-1

## **AGRADECIMIENTOS**

Mi agradecimiento es a mi Dios y Señor por ser mi fortaleza, por consolarme y no desampararme en los momentos difíciles durante el desarrollo de este proyecto y por regalarme la vida para poder disfrutar de este anhelo cumplido.

A mis padres por su amor, apoyo y sacrificio demostrado a lo largo de estos años de estudio. A mis tíos, que son como mis padres, por brindarme su cuidado y comprensión.

Al Director y Codirector de tesis, Ingenieros José Luis Carrillo y Edison Espinosa por su orientación, conocimientos impartidos y ayuda en la elaboración de esta tesis.

A los amigos por los buenos momentos compartidos en el transcurso de esta vida estudiantil universitaria.

*Eithel*

## **DEDICATORIA**

Este proyecto va dedicado exclusivamente a mi Dios y Señor porque quiero entregarle lo mejor de mi vida, mis sueños, logros alcanzados para que en sus manos tengan un propósito y haga de ellos conforme a su voluntad.

*Eithel*

# ÍNDICE

I.- SISTEMAS MULTIAGENTES .....	- 1 -
1.1.- INTRODUCCIÓN .....	- 1 -
1.2.- DEFINICIÓN DE AGENTE.....	- 2 -
1.2.1.- DEFINICIÓN DE ACTOR .....	- 4 -
1.3.- TIPOS DE AGENTES .....	- 5 -
1.3.1.- AGENTES COLABORATIVOS .....	- 5 -
1.3.2.- AGENTES DE INTERFAZ .....	- 5 -
1.3.3.- AGENTES MÓVILES .....	- 5 -
1.3.4.- AGENTES DE INFORMACIÓN (INTERNET) .....	- 6 -
1.3.5.- AGENTES REACTIVOS .....	- 6 -
1.3.6.- AGENTES HÍBRIDOS .....	- 6 -
1.3.7.- AGENTES HETEROGÉNEOS .....	- 7 -
1.4.- ARQUITECTURA DE UN AGENTE .....	- 7 -
1.4.1.- BASADA EN LA LÓGICA.....	- 7 -
1.4.2.- REACTIVA.....	- 8 -
1.4.3.- BDI (BELIEF, DESIRE, INTENTION) .....	- 9 -
1.4.4.- POR CAPAS .....	- 10 -
1.4.5.- DELIBERATIVAS.....	- 10 -
1.4.6.- HÍBRIDAS .....	- 11 -
1.5.- DEFINICIÓN DE SISTEMA MULTIAGENTE (SMA).....	- 11 -
1.5.1.- ORGANIZACIÓN SOCIAL .....	- 12 -
1.5.2.- COORDINACIÓN .....	- 14 -
1.5.3.- COOPERACIÓN.....	- 15 -
1.5.4.- NEGOCIACIÓN .....	- 17 -
1.5.5.- COMUNICACIÓN.....	- 18 -
1.5.5.1.- Métodos de comunicación.....	- 19 -
1.5.5.1.1.- Sistemas de pizarra .....	- 19 -
1.5.5.1.2.- Sistemas de mensajes .....	- 20 -
1.5.5.1.2.1.- Actos de habla .....	- 21 -
1.5.5.2.- Niveles y protocolos de comunicación.....	- 22 -

1.6.- ARQUITECTURA MULTIAGENTE FIPA.....	- 24 -
1.6.1.- SERVICIOS DE DIRECTORIO .....	- 25 -
1.6.2.- GESTIÓN DE AGENTES .....	- 25 -
1.6.2.1.- Plataforma FIPA.....	- 26 -
1.6.3.- TRANSPORTE DE MENSAJES.....	- 28 -
1.6.3.1.- Estructura de un mensaje FIPA ACL .....	- 30 -
1.6.3.2.- Representación de envoltorio .....	- 31 -
1.6.3.2.1.- Parámetros de envoltorio.....	- 31 -
1.6.3.3.- Protocolos de transporte .....	- 32 -
1.6.4.- LENGUAJE DE COMUNICACIÓN FIPA-ACL.....	- 32 -
1.6.4.1.- Lenguaje de contenidos .....	- 33 -
1.6.4.1.1.- Fipa-SL .....	- 33 -
1.6.4.2.- Actos Comunicativos.....	- 33 -
1.6.4.3.- Protocolos de Interacción .....	- 35 -
1.6.5.- SERVICIO DE ONTOLOGÍA.....	- 35 -
1.6.6.- SEGURIDAD DE AGENTES .....	- 36 -
1.6.7.- MOVILIDAD DE AGENTES EN LA PLATAFORMA.....	- 37 -
1.6.8.- NOMBRADO DE AGENTES (AGENT NAME SERVICE-ANS) .....	- 38 -
II.- METODOLOGÍA INGENIAS .....	- 39 -
2.1.- INTRODUCCIÓN .....	- 39 -
2.2.- META – MODELADO.....	- 41 -
2.2.1.- NOMENCLATURA.....	- 41 -
2.2.2.- ENTIDADES BÁSICAS.....	- 42 -
2.2.3.- NOTACIÓN .....	- 44 -
2.3.- META - MODELO DE AGENTE .....	- 46 -
2.3.1.- CONTROL DE AGENTE.....	- 47 -
2.3.1.1.- Agentes en Ejecución .....	- 48 -
2.3.2.- ASOCIACIÓN DE RESPONSABILIDADES.....	- 49 -
2.3.3.- ESTADO MENTAL.....	- 49 -
2.4.- META – MODELO DE INTERACCIÓN .....	- 50 -
2.5.- META – MODELO DE OBJETIVOS Y TAREAS .....	- 52 -
2.6.- META – MODELO DE ORGANIZACIÓN .....	- 54 -



2.6.1.- DESCRIPCIÓN ESTRUCTURAL DE LA ORGANIZACIÓN .....	- 54 -
2.6.2.- DESCRIPCIÓN SOCIAL .....	- 55 -
2.6.2.1.- Prioridad entre las relaciones.....	- 56 -
2.6.3.- DESCRIPCIÓN FUNCIONAL.....	- 57 -
2.7.- META – MODELO DE ENTORNO .....	- 57 -
2.8.- CICLO DE VIDA.....	- 60 -
2.8.1.- GENERACIÓN DE INSTANCIAS DEL META-MODELO DE AGENTE ....	- 63 -
2.8.2.- GENERACIÓN DE INSTANCIAS DEL META-MODELO DE INTERACCIONES.....	- 63 -
2.8.3.- GENERACIÓN DE INSTANCIAS DEL META-MODELO DE TAREAS Y OBJETIVOS .....	- 64 -
2.8.4.- GENERACIÓN DE INSTANCIAS DEL META-MODELO DE ORGANIZACIÓN.....	- 65 -
2.8.5.- GENERACIÓN DE INSTANCIAS DEL META-MODELO DE ENTORNO .-	- 66 -
 III.- PLATAFORMA JADE .....	- 67 -
3.1.- INTRODUCCIÓN .....	- 67 -
3.3.- ENTORNO DE EJECUCIÓN.....	- 71 -
3.3.1.- INSTALACIÓN .....	- 71 -
3.3.2.- EJECUCIÓN .....	- 72 -
3.4.- PROGRAMACIÓN DE AGENTES JADE .....	- 79 -
3.4.1.- CREACIÓN DE AGENTES .....	- 80 -
3.4.2.- COMPILACIÓN Y EJECUCIÓN DE AGENTES .....	- 80 -
3.4.3.- COMPORTAMIENTOS .....	- 81 -
3.4.4.- COMUNICACIÓN ENTRE AGENTES .....	- 83 -
3.4.5.- ONTOLOGÍA.....	- 84 -
3.4.5.1.- Conversión realizada por el soporte de JADE para ontologías. ....	- 84 -
3.4.5.2.- Uso de la herramienta Protégé y BeanGenerator para la creación de ontologías.....	- 85 -
 IV.- DESARROLLO DEL SOFTWARE .....	- 87 -

4.1.- INTRODUCCIÓN .....	- 87 -
4.2.- ANÁLISIS.....	- 88 -
4.2.1.- ESPECIFICACIÓN DE REQUISITOS .....	- 88 -
4.2.1.1.- Introducción.....	- 88 -
4.2.1.1.1.- Propósito.....	- 88 -
4.2.1.1.2.- Ámbito del Sistema .....	- 88 -
4.2.1.1.3.- Definiciones, Acrónimos y Abreviaturas .....	- 89 -
4.2.1.1.3.1.- Definiciones.....	- 89 -
4.2.1.1.3.2.- Acrónimos .....	- 89 -
4.2.1.1.3.3.- Abreviaturas .....	- 89 -
4.2.1.1.4.- Referencias .....	- 89 -
4.2.1.1.5.- Visión General del Documento .....	- 90 -
4.2.1.2.- Descripción General .....	- 90 -
4.2.1.2.1.- Funciones del Sistema .....	- 90 -
4.2.1.2.1.1.- Gestión Administrativa.....	- 90 -
4.2.1.2.1.2.- Gestión de Actividades.....	- 91 -
4.2.1.2.1.3.- Gestión de Reuniones .....	- 92 -
4.2.1.2.2.- Características de los usuarios.....	- 94 -
4.2.1.2.3.- Restricciones.....	- 94 -
4.2.1.2.4.- Dependencias.....	- 94 -
4.2.1.3.- Requisitos Específicos.....	- 94 -
4.2.1.3.1.- Requisitos Funcionales .....	- 95 -
4.2.1.3.1.1.- Gestión Administrativa.....	- 95 -
4.2.1.3.1.2.- Gestión de Actividades.....	- 95 -
4.2.1.3.1.3.- Gestión de Reuniones .....	- 95 -
4.2.1.3.2.- Requisitos de Interfaces Externas.....	- 95 -
4.2.1.3.2.1.- Interfaces de Usuario .....	- 95 -
4.2.1.3.2.2.- Interfaces Hardware.....	- 96 -
4.2.1.3.2.3.- Interfaces Software .....	- 96 -
4.2.1.3.2.4.- Interfaces de Comunicación .....	- 96 -
4.2.1.3.3.- Requisitos de Rendimiento.....	- 96 -
4.2.1.3.4.- Requisitos de Desarrollo.....	- 96 -

4.2.1.3.5.- Requisitos Tecnológicos.....	- 96 -
4.2.1.3.7.- Atributos.....	- 97 -
4.2.1.3.7.1.- Seguridad.....	- 97 -
4.2.1.4.- Anexo Documentos.....	- 98 -
4.2.1.4.1.- Formularios de Entrada.....	- 98 -
4.2.1.4.2.- Formularios de Salida.....	- 99 -
4.2.2.- HERRAMIENTA DE MODELADO DE INGENIAS: IDK (INGENIAS DEVELOPMENT KIT).....	- 100 -
4.2.3.- CASOS DE USO.....	- 101 -
4.2.3.1.- Caso de usos detallados.....	- 102 -
4.2.4.- META-MODELO DE AGENTE.....	- 104 -
4.2.5.- META-MODELO DE INTERACCIONES.....	- 107 -
4.2.6.- META-MODELO DE OBJETIVOS Y TAREAS.....	- 111 -
4.2.7.- META-MODELO DE ENTORNO.....	- 114 -
4.2.8.- META-MODELO DE ORGANIZACIÓN.....	- 116 -
4.2.9.- CONSTRUCCIÓN DE LA ONTOLOGÍA.....	- 117 -
4.3.- DISEÑO.....	- 120 -
4.3.1.- META-MODELO DE AGENTE.....	- 120 -
4.3.2.- META-MODELO DE INTERACCIONES.....	- 121 -
4.3.3.- META-MODELO DE OBJETIVOS Y TAREAS.....	- 123 -
4.3.4.- META-MODELO DE ENTORNO.....	- 123 -
4.3.5.- META-MODELO DE ORGANIZACIÓN.....	- 124 -
4.3.6.- MODELO ENTIDAD RELACION.....	- 131 -
4.4.- IMPLEMENTACIÓN.....	- 132 -
4.4.1.- SELECCIÓN DE LA BASE DE DATOS.....	- 132 -
4.4.2.- GESTIÓN ADMINISTRATIVA.....	- 133 -
4.4.3.- GESTIÓN DE ACTIVIDADES.....	- 133 -
4.4.4.- GESTIÓN DE REUNIONES.....	- 134 -
4.4.5.- SISTEMA OPERATIVO.....	- 135 -
4.4.6.- FUNCIONAMIENTO DEL SISTEMA AMulGeR.....	- 135 -
4.5.- PRUEBAS.....	- 137 -
4.5.1.- CASO 1.....	- 138 -

4.5.2.- CASO 2 .....	- 139 -
4.5.3.- CASO 3 .....	- 141 -
4.5.4.- CASO 4 .....	- 142 -
4.5.5.- CASO 5 .....	- 144 -
4.5.6.- CASO 6 .....	- 144 -
4.5.7.- CASO 7 .....	- 145 -
4.5.8.- CASO 8 .....	- 147 -
4.5.9.- CASO 9 .....	- 147 -
4.5.10.- CASO 10 .....	- 147 -
V.- CONCLUSIONES Y RECOMENDACIONES .....	- 148 -
5.1.- CONCLUSIONES .....	- 148 -
5.2.- RECOMENDACIONES .....	- 149 -
WEBGRAFÍA.....	- 151 -
ANEXO A .....	- 153 -
INTEGRACIÓN DE JADE CON SERVLET (CLASE JADEGATEWAY).....	- 153 -
1.1.- INTRODUCCIÓN A LOS SERVLETS .....	- 153 -
1.2.- INTRODUCCIÓN AL PAQUETE JADE.WRAPPER.GATEWAY .....	- 154 -
1.3.- INTEGRACIÓN DE JADE CON SERVLET .....	- 156 -

## ÍNDICE DE FIGURAS

FIGURA 1.1 ESTRUCTURA DE UN SISTEMA DE PIZARRA.....	- 19 -
FIGURA 1.2 PRINCIPIO DE LA TRANSMISIÓN DE UN MENSAJE.....	- 20 -
FIGURA 1.3 ARQUITECTURA FIPA.....	- 24 -
FIGURA 1.4 PLATAFORMA FIPA.....	- 26 -
FIGURA 1.5 ENVOLTORIO DE UN MENSAJE FIPA .....	- 31 -
FIGURA 1.6 COMPONENTES DEL LENGUAJE DE COMUNICACIÓN FIPA-ACL.....	- 32 -

FIGURA 2.1 RELACIONES ENTRE LOS DIFERENTES META-MODELOS Y LAS DOS ENTIDADES PRINCIPALES, LA ORGANIZACIÓN Y EL AGENTE.....	- 40 -
FIGURA 2.2 NOMENCLATURA PARA LOS NOMBRES DE LAS RELACIONES	- 42 -
FIGURA 2.3 ENTIDADES BÁSICAS DE LA METODOLOGÍA.....	- 43 -
FIGURA 2.4 RELACIONES BÁSICAS DE LA METODOLOGÍA.....	- 43 -
FIGURA 2.5 CONCEPTOS RELEVANTES EN EL CONTROL DEL AGENTE.....	- 47 -
FIGURA 2.6 ELEMENTOS DEL MODELO DE AGENTE.....	- 48 -
FIGURA 2.7 RELACIÓN ENTRE LOS DIFERENTES ASPECTOS DE LA INTERACCIÓN .....	- 51 -
FIGURA 2.8 RELACIONES ENTRE TAREAS Y OBJETIVOS.....	- 53 -
FIGURA 2.9 ELEMENTOS DE DEFINICIÓN DE TAREAS.....	- 53 -
FIGURA 2.10 DESCRIPCIÓN ESTRUCTURAL.....	- 55 -
FIGURA 2.11 DESCRIPCIÓN SOCIAL.....	- 56 -
FIGURA 3.1 ESQUEMA DE DISTRIBUCIÓN DE LOS CONTAINERS Y LAS PLATAFORMAS .....	- 68 -
FIGURA 3.2 PORTABILIDAD DE JADE.....	- 70 -
FIGURA 3.3 INTERFAZ GRÁFICA DEL RMA DE JADE.....	- 73 -
FIGURA 3.4 INTERFAZ GRÁFICA DEL AGENTE DUMMY .....	- 76 -
FIGURA 3.5 INTERFAZ GRÁFICA DEL AGENTE SNIFFER.....	- 78 -
FIGURA 3.6 INTERFAZ GRÁFICA DEL DF GUI.....	- 78 -
FIGURA 3.7 INTERFAZ GRÁFICA DEL AGENTE INTROSPECTOR .....	- 79 -
FIGURA 3.8 FLUJO DE CONTROL DE UN AGENTE BÁSICO: INICIALIZACIÓN, REALIZACIÓN DE LA TAREA Y LIMPIEZA Y FINALIZACIÓN.....	- 82 -
FIGURA 3.9 ESQUEMA DE ENVÍO DE MENSAJES EN JADE.....	- 83 -
FIGURA 3.10 INTERFAZ GRÁFICA DE INSTALACIÓN DE LA HERRAMIENTA PROTEGÉ .....	- 86 -

FIGURA 4.1. INGRESO Y/O MODIFICACIÓN DE UNA ACTIVIDAD.....	- 98 -
FIGURA 4.2. CONVOCAR REUNIÓN Y/O MODIFICAR LUGAR DE UNA REUNIÓN .....	- 98 -
FIGURA 4.3. CONSULTA GENERAL DE ACTIVIDADES Y HORARIOS.....	- 99 -
FIGURA 4.4. CONSULTA INDIVIDUAL DE UNA ACTIVIDAD .....	- 99 -
FIGURA 4.5. CONSULTA GENERAL DE REUNIONES.....	- 99 -
FIGURA 4.6. CONSULTA INDIVIDUAL DE UNA REUNIÓN .....	- 100 -
FIGURA 4.7 EDITOR INGENIAS IDK.....	- 101 -
FIGURA 4.8. CASOS DE USO ASOCIADOS .....	- 101 -
FIGURA 4.9. MODELO DE AGENTE PARA EL AGENTE GESTOR REUNIONES..... .....	- 105 -
FIGURA 4.10 MODELO DE AGENTE PARA EL AGENTE BUSCADOR.....	- 105 -
FIGURA 4.11 MODELO DE AGENTE PARA EL AGENTE AGENDA.....	- 106 -
FIGURA 4.12 MODELO DE AGENTE PARA EL AGENTE INFORMADOR.....	- 106 -
FIGURA 4.13 INTERACCIÓN PARA DETALLAR LA REALIZACIÓN DE CASO DE USO CONVOCAR REUNIÓN .....	- 107 -
FIGURA 4.14 INTERACCIÓN PARA DETALLAR LA REALIZACIÓN DE CASO DE USO MODIFICAR LUGAR .....	- 108 -
FIGURA 4.15 INTERACCIÓN PARA DETALLAR LA REALIZACIÓN DE CASO DE USO CANCELAR PARTICIPACIÓN .....	- 108 -
FIGURA 4.16. INTERACCIÓN PARA DETALLAR LA REALIZACIÓN DE CASO DE USO CANCELAR REUNIÓN .....	- 109 -
FIGURA 4.17 DIAGRAMA DE COLABORACIÓN PARA LA INTERACCIÓN CONVOCAR REUNIÓN.....	- 110 -
FIGURA 4.18 DIAGRAMA DE COLABORACIÓN PARA LA INTERACCIÓN MODIFICAR LUGAR .....	- 110 -

FIGURA 4.19 DIAGRAMA DE COLABORACIÓN PARA LA INTERACCIÓN CANCELAR PARTICIPACIÓN .....	- 110 -
FIGURA 4.20 DIAGRAMA DE COLABORACIÓN PARA LA INTERACCIÓN CANCELAR REUNIÓN .....	- 111 -
FIGURA 4.21 ESTRUCTURACIÓN INICIAL DE OBJETIVOS .....	- 111 -
FIGURA 4.22 TAREAS ASOCIADAS A OBJETIVOS .....	- 112 -
FIGURA 4.23 ASOCIACIONES ENTRE OBJETIVOS Y ROLES .....	- 114 -
FIGURA 4.24 ELEMENTOS IDENTIFICADOS EN EL ENTORNO .....	- 115 -
FIGURA 4.25 REPRESENTACIÓN DEL SISTEMA MULTIAGENTE PARA EL SISTEMA GESTIÓN DE REUNIONES (AMULGER).....	- 116 -
FIGURA 4.26 REFINAMIENTO DEL FLUJO DE TRABAJO GESTIÓN DE REUNIONES.....	- 117 -
FIGURA 4.27 RELACIONES SOCIALES ENTRE AGENTES .....	- 117 -
FIGURA 4.28 UNIDADES DE INTERACCIÓN IDENTIFICADAS PARA LA INTERACCIÓN CONVOCAR REUNIÓN.....	- 121 -
FIGURA 4.29 UNIDADES DE INTERACCIÓN IDENTIFICADAS PARA LA INTERACCIÓN MODIFICAR LUGAR.....	- 122 -
FIGURA 4.30 UNIDADES DE INTERACCIÓN IDENTIFICADAS PARA LA INTERACCIÓN CANCELAR PARTICIPACIÓN .....	- 122 -
FIGURA 4.31 UNIDADES DE INTERACCIÓN IDENTIFICADAS PARA LA INTERACCIÓN CANCELAR REUNIÓN.....	- 122 -
FIGURA 4.32 SATISFACCIÓN DE OBJETIVOS .....	- 123 -
FIGURA 4.33 PERCEPCIÓN DEL AGENTE GESTOR DE REUNIONES.....	- 124 -
FIGURA 4.34 TAREAS QUE COMPONEN EL FLUJO DE TRABAJO MODIFICAR LUGAR.....	- 124 -
FIGURA 4.35 TAREAS QUE COMPONEN EL FLUJO DE TRABAJO CONVOCAR REUNIÓN .....	- 125 -

FIGURA 4.36 TAREAS QUE COMPONEN EL FLUJO DE TRABAJO CANCELAR PARTICIPACIÓN .....	- 126 -
FIGURA 4.37 TAREAS QUE COMPONEN EL FLUJO DE TRABAJO CANCELAR REUNIÓN .....	- 126 -
FIGURA 4.38 DEPENDENCIAS ENTRE LAS TAREAS DEL FLUJO CONVOCAR REUNIÓN .....	- 127 -
FIGURA 4.39 DEPENDENCIAS ENTRE LAS TAREAS DEL FLUJO MODIFICAR LUGAR.....	- 127 -
FIGURA 4.40 DEPENDENCIAS ENTRE LAS TAREAS DEL FLUJO CANCELAR PARTICIPACIÓN .....	- 128 -
FIGURA 4.41 DEPENDENCIAS ENTRE LAS TAREAS DEL FLUJO CANCELAR REUNIÓN .....	- 128 -
FIGURA 4.42 RESPONSABLES DE LA EJECUCIÓN DE TAREAS EN EL FLUJO DE TRABAJO CONVOCAR REUNIÓN .....	- 129 -
FIGURA 4.43 RESPONSABLES DE LA EJECUCIÓN DE TAREAS EN EL FLUJO DE TRABAJO MODIFICAR LUGAR .....	- 129 -
FIGURA 4.44 RESPONSABLES DE LA EJECUCIÓN DE TAREAS EN EL FLUJO DE TRABAJO CANCELAR PARTICIPACIÓN .....	- 130 -
FIGURA 4.45 RESPONSABLES DE LA EJECUCIÓN DE TAREAS EN EL FLUJO DE TRABAJO CANCELAR REUNIÓN .....	- 130 -
FIGURA 4.46 TAREAS QUE PRODUCEN INTERACCIONES .....	- 131 -
FIGURA 4.47 MODELO ENTIDAD RELACIÓN .....	- 132 -
FIGURA A.1 PASOS A SEGUIR EN LA INTEGRACIÓN DE JADE CON SERVLET .....	- 157 -



## ÍNDICE DE TABLAS

TABLA 1.1 ESTRUCTURA DE UN MENSAJE FIPA-ACL.....	- 30 -
TABLA 1.2 ACTOS COMUNICATIVOS DE FIPA .....	- 34 -
TABLA 2.1 NOTACIÓN EMPLEADA EN LA REPRESENTACIÓN DE LOS MODELOS .....	- 45 -
TABLA 2.2 SOLUCIÓN DE JADE PARA EL MODELADO DEL COMPORTAMIENTO Y RESPONSABILIDADES .....	- 47 -
TABLA 2.3 ENUMERACIÓN DE EJEMPLOS DE RECURSOS .....	- 60 -
TABLA 2.4 ASOCIACIONES ENTRE LOS ELEMENTOS DEL RUP Y ENTIDADES DE MAS GRASIA.....	- 61 -
TABLA 2.5 ADECUACIÓN DE LAS ETAPAS DEL RUP A LOS META-MODELOS PRESENTADOS. ....	- 62 -
TABLA 4.1. DEFINICIÓN DE TÉRMINOS .....	- 89 -
TABLA 4.2. ACRÓNIMOS .....	- 89 -
TABLA 4.3. ABREVIATURAS .....	- 89 -
TABLA 4.4 HORARIOS DEL ING. EDISON ESPINOSA EN LOS DÍAS 21-07-2008 Y 22-07-2008.....	- 138 -
TABLA 4.5 HORARIOS DEL ING. RAÚL CAJAS EN LOS DÍAS 21-07-2008 Y 22-07- 2008 .....	- 138 -
TABLA 4.6 HORARIOS DEL ING. RAÚL ROSERO EN LOS DÍAS 21-07-2008 Y 22- 07-2008 .....	- 139 -
TABLA 4.7 HORARIOS DE LA ING. ALEXANDRA CORRAL EN LOS DÍAS 21-07- 2008 Y 22-07-2008 .....	- 139 -
TABLA 4.8 HORARIOS DE LOS ING. JOSÉ LUIS CARRILLO Y EDISON ESPINOSA DEL DÍA 22-07-2008.....	- 140 -
TABLA 4.9 HORARIO DEL ING. ARMANDO ÁLVAREZ DEL DÍA 22-07-2008	- 140 -

TABLA 4.10 HORARIO DEL RECTOR DE LA INSTITUCIÓN DEL DÍA 23-07-2008...	- 141 -
TABLA 4.11 HORARIO DEL ING. ARMANDO ÁLVAREZ DEL DÍA 23-07-2008	- 142 -
TABLA 4.12 HORARIOS DEL RECTOR DE LA INSTITUCIÓN Y DEL ING. JOSÉ LUIS CARRILLO DEL DÍA 24-07-2008	- 143 -
TABLA 4.13 HORARIO DEL ING. ARMANDO ÁLVAREZ DEL DÍA 24-07-2008	- 143 -
TABLA 4.14 HORARIOS DEL SUBDIRECTOR DE LA INSTITUCIÓN Y DEL ING. EDISON ESPINOSA DEL DÍA 24-07-2008	- 145 -
TABLA 4.15 HORARIO DEL SUBDIRECTOR DE DOCENCIA DEL DÍA 24-07-2008	- 145 -
TABLA 4.16 HORARIOS DEL LOS INGS. JOSÉ LUIS CARRILLO Y EDISON ESPINOSA DEL DÍA 23-07-2008	- 146 -
TABLA 4.17 HORARIOS DEL LOS INGS. RAÚL CAJAS Y RAÚL ROSERO DEL DÍA 23-07-2008	- 146 -
TABLA A.1: MÉTODOS DE UN SERVLET	- 154 -

## RESÚMEN

El presente proyecto de investigación está formado por cinco capítulos. En el primer capítulo se estudia el estado del arte de los agentes y sistemas multiagentes (*SMA*), como son definiciones, tipos, arquitecturas, características de los SMA y la arquitectura Multiagente FIPA (*Foundation for Intelligent Physical Agents*).

En el segundo capítulo se estudia la metodología INGENIAS para el modelado de sistemas multiagentes con sus respectivos cinco meta-modelos: agentes, interacción, organización, entorno y objetivos y tareas para aplicarlo en el análisis y diseño del sistema desarrollado en este proyecto.

En el tercer capítulo se estudia el funcionamiento, entorno de ejecución de la plataforma JADE, la construcción de ontologías y la programación de agentes (la comunicación entre ellos, sus comportamientos) para que se ejecuten sobre esta plataforma.

En el cuarto capítulo se detalla todo el proceso de desarrollo de la “Agenda Multiagente orientada a la Web para la Gestión de Reuniones” (AMulGeR), aplicando el conocimiento adquirido en los capítulos anteriores.

En el quinto capítulo se mencionan las conclusiones y recomendaciones obtenidas al finalizar el proyecto de grado.

# CAPÍTULO I

## I.- SISTEMAS MULTIAGENTES

### 1.1.- INTRODUCCIÓN

La Inteligencia Artificial (IA), puede considerarse como una de las disciplinas más nuevas y considerada a la vez como una gran desconocida que más interés despierta. Esto es debido a que pocas personas tienen claro qué es la IA, pero se podría intentar resumirla comentando que la IA trata de *desarrollar sistemas que piensen y actúen racionalmente*.<sup>1</sup>

En los últimos años la IA ha ido evolucionando y ha surgido un nuevo paradigma conocido como *paradigma de agentes*, el cual abarca el desarrollo de entidades que puedan actuar de forma autónoma y razonada, permitiendo abordar de una manera más apropiada la construcción de sistemas inteligentes más complejos aplicados a muy diversos campos.

Si se retoma la definición dada anteriormente donde se consideraba a la IA como un medio para el desarrollo de *sistemas que piensen y actúen racionalmente*, se puede pensar que la IA, en su conjunto, trata de construir dichas entidades autónomas e inteligentes. De acuerdo con esta visión, se puede considerar a la IA como una disciplina orientada a la construcción de *agentes inteligentes* donde se integran las diferentes áreas que ésta comprende.<sup>2</sup>

Los orígenes de esta tecnología comienzan con la Inteligencia Artificial Distribuida (IAD), que integra los conceptos de dos campos de conocimiento: la Inteligencia Artificial y los Sistemas Distribuidos, este último, estudia las propiedades de conjuntos de procesadores autónomos que no comparten memoria primaria, pero sí cooperan comunicándose por medio del envío de mensajes sobre una red de comunicación.<sup>3</sup>

---

<sup>1 2</sup> Agentes Inteligentes el siguiente paso en la Inteligencia Artificial  
[www.ati.es/novatica/2000/145/vjulia-145.pdf](http://www.ati.es/novatica/2000/145/vjulia-145.pdf)

<sup>3</sup> Agentes y Sistemas Multiagentes: [agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm](http://agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm)

Desde que surgió la IAD, se ha interesado, entre otros temas, por estudiar el modelo y comportamiento de varios agentes que cooperan entre sí para la resolución de un problema o desarrollo de una tarea. Los tres ejes fundamentales que se han estudiado en IAD son<sup>4</sup>:

- *Los Sistemas Multiagentes (SMA)*: esta rama de la IAD estudia el comportamiento de agentes inteligentes que resuelven un problema de manera cooperativa.
- *Resolución Distribuida de Problemas (RDP)*: esta rama de la IAD trabaja con las formas de dividir un problema, para asignar las partes a un conjunto de entidades independientes y cooperantes, para que en grupo hallen la solución.
- *La Inteligencia Artificial en Paralelo (IAP)*: esta rama de la IAD se centra en el desarrollo de lenguajes y algoritmos paralelos para sistemas concurrentes en IAD.

Este capítulo del presente proyecto de investigación se centra en la explicación e integración de los conceptos de los SMA, las definiciones, tipos y arquitecturas de los agentes que integran este tipo de sistemas, las características de los SMA que son: organización social, cooperación, coordinación, negociación, comunicación y la arquitectura multiagente FIPA (*“Foundation for Intelligent Physical Agents”*).

## 1.2.- DEFINICIÓN DE AGENTE

Para conocer sobre los sistemas multiagentes, se va a definir primero que es un agente. A continuación se mencionan algunos conceptos:

**[FIPA 97]** (*Foundation for Intelligent Physical Agents*) “... una entidad que reside en un entorno donde recibe datos a través de sensores que reflejan eventos en el entorno y ejecuta comandos que producen efectos en el entorno. Un agente puede ser solamente de software o de hardware. Un agente es el actor fundamental en un dominio y combina uno

---

<sup>4</sup> [Lab et al 93] Labidi, S. y Lejouad, W.; 1993. De l’Intelligence Artificielle Distribuée aux Systèmes Multi-Agents : [agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm](http://agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm)

o más capacidades de servicio en un modelo unificado que puede incluir acceso a software externo, usuarios humanos y comunicación.”<sup>5</sup>

**[Wooldridge 95]** “...un sistema de hardware o (más usualmente) un sistema de computadora basado en software que tiene las siguientes propiedades:

- Autonomía: los agentes operan sin la directa intervención del usuario, y tienen cierto control sobre sus acciones y su estado interno.
- Habilidad social: los agentes interactúan con otros agentes (y posiblemente humanos) usando algún tipo de lenguaje de comunicación de agentes.
- Reactividad: los agentes perciben su entorno (que puede ser el mundo físico, el usuario a través de una interfaz gráfica, una colección de otros agentes, INTERNET, o una combinación de estos), y responden a los cambios que ocurren en él.
- Pro-actividad: los agentes no actúan solamente en respuesta a su entorno, sino que son capaces de exhibir un comportamiento dirigido por objetivos tomando la iniciativa.”<sup>6</sup>

**[IBM]** “Los agentes inteligentes son entidades de software que realizan algunas operaciones en lugar de su usuario o de otro programa con algún grado de independencia y autonomía, empleando algún tipo de conocimiento o representación de los objetivos o deseos del usuario.”<sup>7</sup>

**[Russell 95]** (*AIMA, Artificial Intelligence: a Modern Approach*) “Un agente es cualquier cosa que pueda ser vista como percibiendo su entorno a través de sensores y actuando en ese entorno a través de efectores.”<sup>8</sup>

---

<sup>5</sup> [FIPA 97] FIPA. <http://drogo.csel.it/fipa/> [www.nmis.isti.cnr.it/avancini/FraMaS\\_Tesis.pdf](http://www.nmis.isti.cnr.it/avancini/FraMaS_Tesis.pdf)

<sup>6</sup> [Wooldridge 95] WOOLDRIDGE, M.; JENNINGS, N. Intelligent Agents: Theory and Practice. Knowledge Engineering Review. October, 1995: [www.nmis.isti.cnr.it/avancini/FraMaS\\_Tesis.pdf](http://www.nmis.isti.cnr.it/avancini/FraMaS_Tesis.pdf)

<sup>7</sup> [IBM] <http://activist.gpl.ibm.com/WhitePaper/ptc2.htm>  
[www.nmis.isti.cnr.it/avancini/FraMaS\\_Tesis.pdf](http://www.nmis.isti.cnr.it/avancini/FraMaS_Tesis.pdf)

<sup>8</sup> [Russell 95] RUSSELL, Stuart; NORVIG, Peter. Artificial Intelligence, A Modern Approach. Prentice Hall, 1995: [www.nmis.isti.cnr.it/avancini/FraMaS\\_Tesis.pdf](http://www.nmis.isti.cnr.it/avancini/FraMaS_Tesis.pdf)

[Maes 95] “Los agentes autónomos son sistemas computacionales que habitan en entornos dinámicos complejos, sienten y actúan autónomamente en éste entorno; realizando un conjunto de objetivos o tareas para la cual fueron diseñados.”<sup>9</sup>

Cada agente persigue sus objetivos realizando tareas sobre el entorno que afectarán lo que perciba en el futuro (las tareas tienen efectos sobre el entorno). Finalmente, actúa continuamente sobre un período de tiempo, una vez que un agente fue invocado se ejecuta autónomamente hasta que decide no hacerlo más.<sup>10</sup>

Análogamente, un *programa* de computadoras en un entorno real, por ejemplo un sistema bancario, puede ser visto como que percibe el mundo a través de sus entradas y actúa a través de sus salidas. Pero no es un agente, porque sus salidas podrían no afectar qué va a percibir en el futuro, ya que no realiza un cambio en el estado del mundo; además se ejecuta una vez y después deja de hacerlo hasta que es llamado otra vez, mientras que los agentes, una vez iniciados, se ejecutan autónomamente. Todos los agentes son programas, pero no todos los programas son agentes.

### 1.2.1.- DEFINICIÓN DE ACTOR

Por otro lado, está el concepto de *actor*. Un actor es una entidad que procesa los mensajes entrantes a la casilla de correo, cuya dirección nombra al actor.

La reacción de un actor ante la llegada de un mensaje es determinada por su comportamiento (definido en un script que puede modificarse en su ciclo de vida). El modelo de actor ha sido ampliamente usado en la programación orientada a objetos concurrente y comparte con los agentes las características de identidad, comunicación y coordinación. Sin embargo, los agentes poseen atributos tales como: autonomía (para decidir que acción ejecutar en el próximo instante de tiempo), adaptabilidad (a

---

<sup>9</sup> [Maes 95] MAES, P. Artificial Life Meets Entertainment: Life like Autonomous Agents. Communications of the ACM, 38 (11). 1995: [www.nmis.isti.cnr.it/avancini/FraMaS\\_Tesis.pdf](http://www.nmis.isti.cnr.it/avancini/FraMaS_Tesis.pdf)

<sup>10</sup> FraMaS Un Framework para Sistemas Multi-agente: [www.nmis.isti.cnr.it/avancini/FraMaS\\_Tesis.pdf](http://www.nmis.isti.cnr.it/avancini/FraMaS_Tesis.pdf)

través del aprendizaje) y movilidad. Los actores no son agentes pero pueden ser el soporte de implementación de agentes.

### **1.3.- TIPOS DE AGENTES**

#### **1.3.1.- AGENTES COLABORATIVOS**

Los agentes colaborativos enfatizan en la autonomía y la cooperación para ejecutar tareas por ellos mismos, pueden aprender, pero este no es un aspecto típico de su énfasis.

Están enfocados a solucionar problemas muy grandes para un solo agente centralizado; permiten la interconexión de sistemas expertos, sistemas basados en el conocimiento, sistemas tradicionales y proveer soluciones a problemas inherentemente distribuidos (control de tráfico aéreo).

#### **1.3.2.- AGENTES DE INTERFAZ**

Los agentes de interfaz trabajan sobre el aprendizaje y la autonomía. Su objetivo es el de servir a las necesidades del usuario con base en el análisis de sus hábitos y comportamientos, ajustándose de acuerdo a las decisiones anteriormente tomadas.

Este tipo de agentes aprenden a asistir a su usuario observando e imitando las acciones del mismo; la retroalimentación positiva o negativa del usuario sobre una acción tomada de manera autónoma por el agente y la recepción explícita de instrucciones del usuario.

#### **1.3.3.- AGENTES MÓVILES**

Los agentes móviles son capaces de salir de los límites de las redes locales y acceder a computadores en redes remotas (redes WAN) o incluso Internet. De esta forma pueden hacer un recorrido interactuando en cada sitio con otros agentes en nombre de su usuario a



fin de satisfacer su(s) objetivo(s), para luego retornar los resultados al usuario origen. Entre algunos ejemplos tenemos: búsqueda de información, reservas de pasajes, etc.

#### **1.3.4.- AGENTES DE INFORMACIÓN (INTERNET)**

Los agentes de información surgen de la necesidad de recolectar, administrar y clasificar grandes volúmenes de información, como es el Internet. La labor que realiza es filtrar la información de manera autónoma y sin intervención del usuario desde múltiples fuentes. Este tipo de agentes pueden ser estáticos (similares a los agentes de interfaz) o móviles, no cooperativos o sociales y pueden o no aprender, de allí que no haya un modelo estándar que defina su modo de operación.

#### **1.3.5.- AGENTES REACTIVOS**

Los agentes reactivos son una clase especial de agentes, que actúan de manera reactiva a un evento o estímulo producido dentro del entorno del sistema, ya que no poseen internamente un modelo del entorno en el que se encuentran. Por si mismos no agregan mucha autonomía y normalmente son conjuntos de agentes los que realizan labores autónomas.

Hay un número relativamente pequeño de aplicaciones software basadas en agentes reactivos, por esta razón no hay un modo estándar para su operación. Se los puede utilizar para simular muchos tipos de mundos artificiales así como también fenómenos naturales.

#### **1.3.6.- AGENTES HÍBRIDOS**

Los agentes híbridos son aquellos que están formados por la combinación de dos o más características de los agentes mencionados anteriormente para formar uno solo.

Usualmente las arquitecturas híbridas aprovechan los beneficios de poder reaccionar rápidamente a eventos percibidos del mundo exterior y emplean al menos una sección deliberativa (*con un modelo interno de razonamiento*).

### **1.3.7.- AGENTES HETEROGÉNEOS**

Los sistemas de agentes heterogéneos permiten que varios programas que proporcionan una amplia gama de servicios y que trabajan por separado, puedan interoperar entre sí para la solución de un problema, por medio de la integración de diferentes clases de agentes que además pueden contener una o más clases de agentes híbridos.

En base a estos sistemas ha surgido la “*Ingeniería del Software basada en Agentes*” que intenta facilitar la operación entre sistemas de agentes heterogéneos. Uno de los requerimientos principales para este tipo de sistemas es disponer de un Lenguaje de Comunicación entre Agentes (*LCA, ó ACL: Agent Communication Language*).<sup>11</sup>

### **1.4.- ARQUITECTURA DE UN AGENTE**

La arquitectura determina los mecanismos que utiliza un agente para reaccionar a los estímulos, actuar, comunicarse, etc. Además especifican, cómo se descomponen los agentes en un conjunto de módulos que interactúan entre sí para lograr la funcionalidad requerida. A continuación se presenta algunas de ellas.

#### **1.4.1.- BASADA EN LA LÓGICA**

El comportamiento inteligente del agente se genera dándole al sistema una representación simbólica del entorno y del comportamiento deseado.

Esta representación simbólica del entorno se consigue mediante fórmulas lógicas, y la manipulación sintáctica, a través de una deducción lógica o demostración de teoremas. Por tal motivo en los agentes con este tipo de arquitectura, se asume que el estado es una base de datos de fórmulas de predicados de lógica de primer orden. La base de datos es la información que tiene el agente sobre el entorno. El proceso de toma de decisiones del

---

<sup>11</sup> [Witting 92] WITTING, T. ARCHON: An Architecture for Multi-agent Systems. London: Ellis Horwood. 1992 : [www.nmis.isti.cnr.it/avancini/FraMaS\\_Tesis.pdf](http://www.nmis.isti.cnr.it/avancini/FraMaS_Tesis.pdf)

agente se modela a través de un conjunto de reglas de deducción (reglas de inferencia). Así, el comportamiento del agente queda determinado por las reglas de deducción y la base de datos actual.

Se suele utilizar los siguientes tipos de lógicas: Lógica proposicional y de predicados, Lógica Modal, Lógica deóntica, Lógica dinámica, Lógica temporal.

#### **1.4.2.- REACTIVA**

Los agentes con este tipo de arquitectura no obtienen su inteligencia de modelos internos sino de su interacción con su entorno. Se caracterizan principalmente por no tener como elemento central de razonamiento un modelo simbólico y por no utilizar razonamiento simbólico complejo. Basta con que observe su entorno y reconozca una serie de principios simples o dependencias.<sup>12</sup>

Su arquitectura se basa en un sistema simple de *estímulo/respuesta* y está conformada por sensores, módulos de competencia y actuadores.

Los *sensores* recogen la información de su entorno y reconocen la ocurrencia de situaciones cambiantes, la envían a los *módulos de competencia* correspondientes encargados de una tarea específica no compleja (normalmente se hace en un formato de texto plano, es decir no se utilizan representaciones simbólicas o lenguajes de alto nivel), produciéndose una reacción como salida en los mismos, que se transmite al exterior por medio de *actuadores*

La mayor aplicación de este tipo de arquitecturas se ha centrado en el desarrollo de controladores en robótica. Los robots se pueden considerar como agentes reales (no software) que actúan en un entorno cambiante.

---

<sup>12</sup> [Brooks R. A] (1990) 'Elephants Don't Play Chess', DESIGNING AUTONOMOUS AGENTS: Theory and Practice From Biology to Engineering and Back, pp 3-17, Maes, Pattie, (Ed), Cambridge, MA, The MIT Press, 1990: [postgrado.usal.es/bisite/images/stories/publicaciones/otras/2004/c1.pdf](http://postgrado.usal.es/bisite/images/stories/publicaciones/otras/2004/c1.pdf)

### 1.4.3.- BDI (BELIEF, DESIRE, INTENTION)

La arquitectura BDI está caracterizada por el hecho de que los agentes que la implementan están dotados de los estados mentales de *Creencias, Deseos e Intenciones*<sup>13</sup>. A más de estos componentes, forman parte de la arquitectura, las funciones que representan su deliberación y el razonamiento de fines y medios.

Las *creencias* vienen a ser la base de datos de hechos y reglas, la base del conocimiento sobre el entorno en que se desenvuelve. Los *deseos* son las metas finales a alcanzar, en función de ellos se generan las opciones o comportamientos a seguir para lograr los objetivos. Y las *intenciones* es el conjunto de acciones sin ejecutar como producto del sistema de creencia y deseos del agente.

El razonamiento práctico involucra dos procesos importantes: decidir qué metas se desean conseguir, proceso que se conoce como deliberación; y cómo se lograrán estas metas, procedimiento que se denomina razonamiento de fines y medios. El proceso de decisión empieza típicamente tratando de comprender qué opciones están disponibles; una vez generado este conjunto de alternativas, se debe elegir entre ellas y comprometerse con una; esta opción escogida se convierte en una intención, la cual determina las acciones del agente. Las intenciones localizan el razonamiento práctico futuro del agente; cuando se tiene una intención en particular, se descartan todas aquellas opciones que sean inconsistentes con la intención. Además una vez adoptada una intención, el agente debe perseverar en ésta, sólo debe rectificarla cuando la razón por la cual tenía la intención ha cambiado; o cuando el agente sabe con certeza que no podrá cumplir con ella. Finalmente, las intenciones están estrechamente relacionadas con las creencias acerca del futuro. Cuando tiene una intención, el agente al menos debe creer que tiene una gran posibilidad de cumplir con ella.

---

<sup>13</sup> [Haddadi A. y Sundermeyer K]. (1996) “Belief-Desire-Intention Agent Architectures”, Foundations of Distributed Artificial Intelligence, pp 169-185, O’Hare G. M. P.; Jennings N. R. (Eds) Wiley- Interscience Publication. 1996: [postgrado.usal.es/bisite/images/stories/publicaciones/otras/2004/c1.pdf](http://postgrado.usal.es/bisite/images/stories/publicaciones/otras/2004/c1.pdf)

#### **1.4.4.- POR CAPAS**

Dado el requerimiento que un agente sea capaz de tener comportamientos reactivos y pro-activos, una descomposición clara es crear subsistemas separados para tratar estos tipos de comportamientos diferentes. Por lo tanto en esta arquitectura, la toma de decisiones está particionada en un número de capas, cada una de las cuales trata con el entorno del agente a diferente nivel de abstracción.

Típicamente habrá dos capas para tratar con el comportamiento reactivo y pro-activo, respectivamente, sin embargo cuantas más capas haya, más útil es la topología de tales arquitecturas, por el flujo de información y control que hay entre ellas. En general, se puede identificar dos tipos de flujo de control:

- *Capas horizontales*: cada capa de software está directamente conectada a las entradas sensoriales y a las salidas actuadoras. En efecto cada capa actúa por si misma, como un agente, produciendo sugerencias de qué acción realizar
- *Capas verticales*: Una capa se encarga de manipular las entradas sensoriales y las salidas actuadoras

La desventaja de las arquitecturas en capas horizontales es que las capas están cada una compitiendo con las otras para generar la acción sugerida, existe el peligro de que el comportamiento global del agente no sea coherente. Para asegurar que las arquitecturas en capas horizontales sean consistentes, generalmente incluyen una función de mediación, la cual decide qué capa tiene el “control” del agente en un instante determinado.

#### **1.4.5.- DELIBERATIVAS**

En los agentes con este tipo de arquitectura las decisiones se toman bajo mecanismos de razonamiento lógico utilizando modelos de representación simbólica del mundo basados en la correspondencia de patrones y la manipulación simbólica, con el propósito de alcanzar los objetivos del agente.

Esta arquitectura suele basarse en la teoría clásica de planificación de inteligencia artificial: dado un estado inicial, un conjunto de operadores/planes y un estado objetivo, la deliberación del agente consiste en determinar qué pasos debe encadenar para lograr su objetivo, siguiendo un enfoque descendente.

Se puede distinguir los siguientes tipos principales de arquitecturas deliberativas o simbólicas: arquitecturas intencionales y arquitecturas sociales.

- Los *agentes intencionales* se distinguen por ser capaces de razonar sobre sus creencias e intenciones. Se pueden considerar como sistemas de planificación que incluyen creencias e intenciones en sus planes.
- Los *agentes sociales* se pueden definir como agentes intencionales que mantienen además un modelo explícito de otros agentes y son capaces de razonar sobre estos modelos.

#### **1.4.6.- HÍBRIDAS**

Las arquitecturas híbridas pretenden combinar aspectos de deliberativos y reactivos. Una primera propuesta puede ser construir un agente compuesto de dos subsistemas: uno deliberativo, que utilice un modelo simbólico y que genere planes en el sentido expuesto anteriormente, y otro reactivo centrado en reaccionar a los eventos que tengan lugar en el entorno y que no requiera un mecanismo de razonamiento complejo.

#### **1.5.- DEFINICIÓN DE SISTEMA MULTIAGENTE (SMA)**

En general los SMA tratan sobre la coordinación inteligente entre una colección de 'agentes' autónomos, cómo pueden coordinar sus conocimientos, metas, propiedades y planes para tomar una decisión o resolver un problema<sup>14</sup>.

---

<sup>14</sup> Bond A.H., Gasser L. (1988) Readings in Distributed Artificial Intelligence. Morgan Kaufman postgrado.usal.es/bisite/images/stories/publicaciones/otras/2004/c1.pdf

Dentro de la terminología de este campo es importante clarificar en primer lugar la diferencia entre un sistema basado en agentes y un sistema multiagente. Un sistema basado en agentes es aquel que utiliza el concepto de agente como mecanismo de abstracción, pero aunque sea modelado en términos de agentes podría ser implementado sin ninguna estructura de software correspondiente a éstos. Por otro lado, un sistema multiagente es aquel que se diseña e implementa pensando en que estará compuesto por varios agentes que interactuarán entre sí, de forma que juntos permitan alcanzar la funcionalidad deseada. En este caso, hay que hacer un mayor esfuerzo de abstracción, identificar mecanismos de aprendizaje, coordinación, negociación, etc. Los sistemas multiagente son adecuados para solucionar problemas para los que hay múltiples métodos de resolución y/o múltiples entidades capaces de trabajar conjuntamente para solucionarlos.<sup>15</sup>

Por ello, uno de los aspectos básicos en estos sistemas es la organización social de los agentes que lo forman y la interacción entre ellos, es decir la coordinación, cooperación, negociación y comunicación entre los mismos.

### **1.5.1.- ORGANIZACIÓN SOCIAL**

Es la manera como el grupo de agentes está constituido en un instante dado. La organización social está relacionada con la estructura de los componentes funcionales del sistema, sus características, sus responsabilidades, sus necesidades y la manera cómo realizan sus comunicaciones. Esta organización puede ser estática o dinámica, dependiendo de las funciones o tareas de cada agente.

Se puede considerar que una sociedad de agentes está constituida por tres elementos:

- Un grupo de agentes.
- Un conjunto de tareas a realizar.
- Un conjunto de recursos.

---

<sup>15</sup> AGENTES Y SISTEMAS MULTIAGENTE  
[www.ceditec.etsit.upm.es/index.php/Descargar-documento/3-Agentes-y-Sistemas-Multiagente.html](http://www.ceditec.etsit.upm.es/index.php/Descargar-documento/3-Agentes-y-Sistemas-Multiagente.html)

La realización de las tareas por parte de los agentes, puede ser organizada de varias formas, por ejemplo: cada agente ejecuta una de las tareas, o bien, las tareas son divididas en subtareas, por medio de algún mecanismo de descomposición de problemas y estas subtareas son aquellas realizadas por los agentes. Las tareas que debe realizar un agente dependen, entre otros factores, del rol que este agente asume en el sistema. Para la realización de tareas un agente puede necesitar recursos del sistema, en este caso tiene que coordinarse con los otros agentes del sistema que deseen usar el mismo recurso.

La organización en los SMA depende del tipo de comunicación y el modo de cooperación entre agentes, así como del tipo de agentes que conforman el grupo.

En general se pueden distinguir tres tipos de configuraciones organizacionales:

- Estructura Centralizada: En este tipo de configuración existe un agente que controla la interacción de los demás agentes del sistema, porque tiene la información o la funcionalidad para hacerlo.
- Estructura Horizontal: Este tipo de configuración existe cuando todos los agentes que integran un sistema están al mismo nivel, es decir, no hay ningún agente que haga las veces de maestro o supervisor, ni tampoco agentes esclavos.
- Estructura Jerárquica: Esta configuración existe cuando los agentes trabajan diferentes niveles de abstracción de un problema, es decir, la configuración es de niveles. En un mismo nivel se establece una configuración horizontal, si hay más de un agente. Para resolver un problema cada agente divide el problema en subproblemas que él puede resolver con la cooperación de los agentes que están al mismo nivel y subproblemas que sabe que los agentes, de niveles inferiores de la jerarquía, pueden resolver.
- Estructura "ad hoc": Esta configuración puede ser una mezcla de las tres anteriores, se caracteriza porque la dinamicidad de la estructura está regida por el ajuste mutuo entre los pequeños grupos de agentes en el sistema.<sup>16</sup>

---

<sup>16</sup> [Lab et al 93] Labidi, S. y Lejouad, W. ; 1993. De l'Intelligence Artificielle Distribuée aux Systèmes Multi-Agents: [agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm](http://agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm)



Escoger una u otra estructura de organización depende de las funciones que deben cumplir los agentes del sistema, de las características de ellos y de qué tan complejo se quiere el sistema.

### 1.5.2.- COORDINACIÓN

Cuando el número de agentes del sistema crece, aparece la necesidad de ayuda para localizar otros agentes que tengan la información o que nos puedan ofrecer los servicios que requerimos<sup>17</sup>. Por lo tanto la coordinación permite a un agente obtener información y resultados de otros agentes que únicamente ellos pueden proporcionarle. Mediante la coordinación también es posible reducir el coste aprovechando la sinergia de las acciones de varios agentes.

Se definen cuatro tipos de coordinación en los sistemas multiagentes: <sup>18</sup>

*Sincronización de acciones.*- Este tipo de coordinación se centra en la definición de tiempo de ejecución para las diversas tareas, de manera que se ejecuten con una determinada temporización. Son técnicas de alta rapidez, nada adaptables, con casi ninguna capacidad de predicción, se pueden utilizar en sistemas centralizados y distribuidos, y utilizan mensajes para su comunicación. Aunque limitan la libertad de acción, evita conflictos y permite un elevado número de agentes.

*Planificación.*- El fundamento de esta técnica es la subdivisión de tareas en subtareas y su distribución, pero ha de tenerse en cuenta la capacidad de replanificar y la interacción entre los planes de los agentes. Son técnicas de baja rapidez, casi nada adaptables, con una alta capacidad de predicción, se pueden utilizar en sistemas centralizados y distribuidos, utilizan mensajes para su comunicación, la libertad de acción está limitada, evita

---

<sup>17</sup> [Omicini, 2001] A. Omicini, F. Zambonelli, M. Klush y R. Tolksdorf (eds.), Coordination of Internet Agentes. Models, Technologies, and Applications, Springer-Verlag, 2001  
[www.dccia.ua.es/dccia/inf/asignaturas/AIW/docs/ACA.pdf](http://www.dccia.ua.es/dccia/inf/asignaturas/AIW/docs/ACA.pdf)

<sup>18</sup> [Ferber, J., (1999)] Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence, Addison-Wesley: [postgrado.usal.es/bisite/images/stories/publicaciones/otras/2004/c1.pdf](http://postgrado.usal.es/bisite/images/stories/publicaciones/otras/2004/c1.pdf)

conflictos, permite un bajo número de agentes, la cantidad de datos intercambiados es muy alta y es muy complicada de implementar.

La planificación puede verse, en un entorno multiagente, como tres pasos interrelacionados: pensar el plan, distribuir y coordinar las distintas acciones del plan y ejecutar las acciones. Si el primer paso lo realiza un agente, se habla de *planificación centralizada*, si cada agente construye su propio subplan entonces se habla de *planificación distribuida*. Si existe un agente que coordina los distintos subplanes se habla de coordinación *centralizada de planes parciales*.

Coordinación reactiva.- La coordinación entre agentes reactivos se lleva a cabo mediante dos técnicas: la definición de un comportamiento reactivo que favorezca la coordinación, y la inclusión de marcas en el entorno que sean captadas por otros agentes. La primera de las técnicas utiliza la observación de la naturaleza para descubrir qué reglas de comportamiento muy sencillas, que se encuentran en todos los agentes de una sociedad, hacen que el conjunto de los agentes realice una tarea compleja. La segunda de las técnicas consiste en el marcado del entorno por parte de un agente para que esa marca afecte al resto de los agentes reactivos. Las marcas se utilizan fundamentalmente para la sincronización de acciones entre agentes reactivos

### **1.5.3.- COOPERACIÓN**

En un SMA existen dos tipos de tareas que deben ser realizadas: las tareas *locales* y las *tareas* globales. Las tareas locales son aquellas relacionadas con los intereses individuales de cada agente y las tareas globales son aquellas relacionadas con los intereses globales del sistema. Estas tareas globales son descompuestas y cada subtarea es realizada por un agente, de acuerdo a sus habilidades y bajo el supuesto de que la integración de la solución de las subtareas, llevará a la solución global. La descomposición de la tarea global no necesariamente garantiza la independencia de cada una de las subtareas, por ello se necesitan mecanismos de cooperación que permitan compartir resultados intermedios que lleven al progreso en la resolución de las tareas de otros agentes y al progreso de la solución global que debe alcanzar el sistema.

Para que los agentes puedan cooperar de manera eficiente, cada uno de ellos debe tener ciertas características:

- Tener un modelo bien definido del mundo, que le permite localizar a los demás agentes, saber cómo comunicarse con ellos, qué tareas pueden realizar, etc.
- Poder integrar información de otros agentes con la suya, para formar conceptos globales o conocimiento conformado por varios agentes.
- Poder interrumpir un plan que se esté llevando a cabo para ayudar o atender a otros agentes para que puedan cooperar entre sí cuando los agentes lo necesiten.

La cooperación depende mucho de la configuración organizacional del grupo de agentes. Si la estructura es centralizada los agentes dependientes piden colaboración casi permanente al agente maestro, si la estructura es jerárquica, la cooperación puede hacerse por niveles (en un mismo nivel) o de niveles superiores a niveles inferiores y si la estructura es horizontal la cooperación se hace entre todos los agentes.

Existen varios modelos de cooperación, dentro de los cuales se mencionan:

- Cooperación compartiendo tareas y resultados: Los agentes tienen en cuenta las tareas y los resultados intermedios de los demás para realizar las tareas propias.
- Cooperación por delegación: Un agente supervisor o maestro descompone una tarea en subtareas y las distribuye entre los agentes esclavos, para que sean resueltas. Después, el supervisor integra las soluciones para hallar la solución al problema inicial.
- Cooperación por ofrecimiento: Un agente maestro descompone una tarea en subtareas y las difunde en una lista a la que tienen acceso los agentes que integran el sistema, esperando que ellos ofrezcan su colaboración de acuerdo a sus habilidades. El supervisor escoge entre los ofrecimientos y distribuye las subtareas.

#### 1.5.4.- NEGOCIACIÓN

Para que los mecanismos de cooperación y coordinación sean exitosos en un sistema de agentes que actúan interdependientemente, debe existir un mecanismo adicional, por medio del cual, los integrantes de un sistema se puedan poner de acuerdo cuando cada agente defiende sus propios intereses, llevándolos a una situación que los beneficie a todos teniendo en cuenta el punto de vista de cada uno. Este mecanismo es llamado negociación<sup>19</sup>.

Los procesos de negociación tienen como resultado la modificación o confirmación de las creencias de cada agente involucrado, en lo relacionado con los demás agentes y con el mundo en el que se desenvuelve.

Una clasificación interesante desde el punto de vista individual del agente divide en cuatro las posibles formas de actuar en la negociación:

- *Cooperación simétrica*: La negociación produce resultados mejores que los que cada agente obtendría de manera individual.
- *Compromiso simétrico*: Los agente logran el objetivo independientemente y negocian un compromiso entre ambas partes, de manera que se degrada el resultado en comparación con las soluciones independientes. Es esencial un compromiso debido a que no se puede obviar a ninguno de los agentes.
- *Cooperación/compromiso no simétrico*: Uno de los agentes obtiene una mejora de su resultado, mientras que el otro empeora su solución.
- *Conflicto*: No se puede llegar a un acuerdo razonable debido a que los objetivos entran en conflicto. La negociación debe terminar sin tener un resultado claro.

La negociación se caracteriza por tener los siguientes elementos:

- Un número adecuado de agentes involucrados en el proceso.

---

<sup>19</sup> [Her95] Hernández, Marcela.; 1995. Modelaje de Procesos de Negociación como Sistemas de Comunicación: [agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm](http://agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm)

- Un conjunto mínimo de acciones que se llevan a cabo en el proceso, como: proponer, evaluar, refutar, contraproponer, aceptar, rechazar, modificar, etc. Este conjunto es llamado lenguaje: "El principal componente de la negociación como actividad social es el lenguaje". Este conjunto de acciones puede ser visto como un conjunto de actos de habla con una lógica y una semántica especial.

Para que una negociación sea exitosa es necesario un protocolo que facilite y en lo posible garantice la convergencia de ideas a una solución común. Un protocolo establece un conjunto de pasos que debe seguir un proceso de negociación, así como las posibles respuestas de un agente, a las acciones de otro agente.

Existen distintas técnicas de negociación: votación (voting), subasta (auction), pacto (bargaining), mecanismos de mercado (market mechanisms), contratación (contracting) y coaliciones (coalition formation).

#### **1.5.5.- COMUNICACIÓN**

La comunicación entre los agentes les permite sincronizar acciones, enviar y recibir conocimiento, resolver conflictos en la resolución de una tarea, etc. La comunicación es el soporte de la organización, cooperación, coordinación y de los procesos de negociación que se dan entre los agentes.<sup>20</sup>

El proceso de comunicación permite que los agentes desarrollen sus acciones con una visión menos local, tras un proceso de sincronización con el resto de los agentes. Sin embargo, el problema que surge cuando dos seres inteligentes intentan interactuar es que necesitan unas normas y un canal de comunicación, deben utilizar el mismo lenguaje, estar de acuerdo en el significado de los símbolos de ese lenguaje, tener un mecanismo de comunicación para el intercambio de mensajes, no hablar al mismo tiempo, etc.

---

<sup>20</sup> ACA. ARQUITECTURAS Y COMUNICACIÓN ENTRE AGENTES  
[www.dccia.ua.es/dccia/inf/asignaturas/AI/docs/ACA.pdf](http://www.dccia.ua.es/dccia/inf/asignaturas/AI/docs/ACA.pdf)

### 1.5.5.1.- Métodos de comunicación

Los métodos de comunicación se pueden diferenciar en sistemas de pizarra y sistemas de mensaje/diálogo.

#### 1.5.5.1.1.- *Sistemas de pizarra*

La pizarra (blackboard) es una estructura de datos que es usada como una zona de trabajo común que permite a los agentes intercambiar información, datos y conocimiento y es gestionada y arbitrada por un controlador.

El funcionamiento de este sistema de pizarra es de la siguiente manera: un agente inicia una comunicación escribiendo algún tipo de información en la pizarra; en ese momento estos datos están disponibles para todos los agentes del sistema; cada agente accederá a la pizarra eventualmente para comprobar si hay información nueva; normalmente cada agente no recogerá toda la información que se vaya escribiendo en la pizarra, sino que sólo obtendrá aquella que le interese.

Como se observa en la figura 1.1 no existe una comunicación directa entre agentes, cada agente debe resolver su problema de forma autónoma. La estructura central de este sistema es un obstáculo para sistemas orientados a redes, puesto que la sola lectura de la información contenida en la pizarra representa un cuello de botella. En un sistema multiagente se puede disponer de varias pizarras con distintos agentes registrados en cada una.

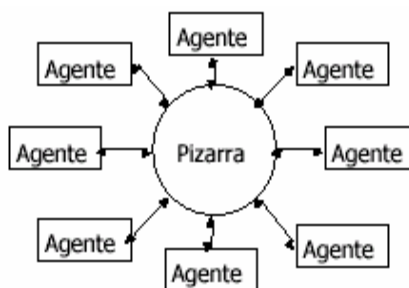


Figura 1.1 Estructura de un Sistema de Pizarra

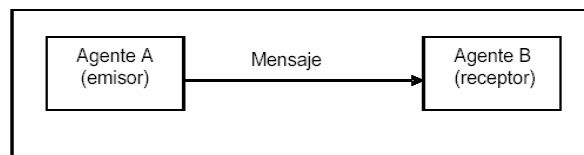
En los sistemas más avanzados incorporan nuevos conceptos para coordinar el mantenimiento de la pizarra:

- Moderador. Agente especializado con conocimiento de control y de evaluación que publica en la pizarra los subproblemas a resolver y decide a qué agentes se asignan de entre aquellos que se han ofrecido a resolverlos.
- Despachador. Agente que informa a los agentes afectados por algún cambio producido en la pizarra.

Los sistemas de pizarra constituyen un método muy flexible de comunicación para la resolución distribuida de problemas. Son independientes de la estrategia de cooperación que se vaya a utilizar y no afectan a la arquitectura de los agentes individuales.

#### 1.5.5.1.2.- *Sistemas de mensajes*

Este método de comunicación muy flexible se establece directamente entre dos agentes *emisor* y *receptor*. El primero transfiere un mensaje específico a otro agente, el *receptor* (Figura 1.2). Al contrario que en los sistemas de pizarra, los mensajes son intercambiados directamente entre dos agentes. No se utiliza memoria, ni otros agente son capaces de leer el mensaje si no va dirigido a ellos. Aunque en las situaciones normales el emisor asigna una única dirección a un mensaje, podemos utilizar el “broadcasting” como una excepción, en la cual el mensaje es enviado a todos los agentes del sistema o a un grupo específico.



**Figura 1.2 Principio de la transmisión de un mensaje**

En este método, los mensajes que los agentes intercambian con otros agentes pueden ser utilizados para establecer comunicaciones y mecanismos de cooperación utilizando protocolos definidos. Es necesario establecer protocolos de comunicación que especifiquen

el proceso de comunicación, el formato de los mensajes y el lenguaje de comunicación. Todos los agentes deben conocer la semántica del lenguaje de comunicación.

Realmente el contenido de un mensaje depende del estado mental de los agentes involucrados (emisor y receptor) y de la relación entre ambos. Un mensaje producirá cambios inicialmente en el estado mental del receptor y eventualmente en el resto del entorno. La teoría de los actos de habla intenta atender estas cuestiones.

#### ***1.5.5.1.2.1.- Actos de habla***

Debido a que la mayoría de los actos comunicativos entre personas se realizan mediante el lenguaje hablado, los expertos en lingüística utilizan el término *Acto del Habla* para referirse a todos los tipos de actos comunicativos<sup>21</sup>.

La comunicación entre agentes está basada en esta teoría, la cual trata las comunicaciones o actos de habla como acciones. Numerosos de estos actos comunicativos o de habla no sirven únicamente para constatar un hecho (verdadero o falso) o describirlo sino que lo realizan. Estos actos comunicativos se los denomina “*performatives*”. Habitualmente se traduce por performativa.

*Ejemplo:* Quien tras pisar a otro le dice “le pido disculpas”, no está enunciando un hecho verdadero o falso está realizando la acción de pedir disculpas.

En un Acto de Habla se distingue entre:

- *Locución:* hecho físico o contenido proposicional del acto de habla.
- *Ilocución:* intención con que se realiza, transferencia de intenciones (el que habla quiere que el que escucha haga algo o piense algo como consecuencia de sus palabras).
- *Perlocución:* acciones que ocurren como resultado de la ilocución.

---

<sup>21</sup> [Searl,2001] John R. Searle, *Mente, lenguaje y Sociedad. La Filosofía en el Mundo real*, Alianza editorial, 2001: [www.dccia.ua.es/dccia/inf/asignaturas/AI/docs/ACA.pdf](http://www.dccia.ua.es/dccia/inf/asignaturas/AI/docs/ACA.pdf)



Ejemplo: “Abre el grifo”

- *Locución:* la frase “abre el grifo”, el que la dice, el que la escucha y el objeto es el grifo de agua
- *Ilocución:* yo quiero que el escuchante abra el grifo
- *Perlocución:* Si el escuchante abre o no el grifo

Los posibles actos de habla se clasifican en cinco clases:

- *Asertiva:* El emisor del mensaje realiza una aserción, es decir informa de la verdad o falsedad de la proposición
- *Compromisoria:* El hablante se compromete con una acción futura que depende de él. Variantes pueden ser promesa, amenaza.
- *Directiva:* El hablante enuncia una acción futura del oyente, instándole de alguna forma a que la realice. Son las órdenes, peticiones, sugerencias, etc.
- *Declarativa:* El hablante enuncia una acción presente suya, que se realiza por el mero enunciado. Son la disculpa, aprobación
- *Expresiva:* Corresponde típicamente a las oraciones exclamativas. El hablante expresa mediante ellas sus estados de ánimo. Por ejemplo: agradecimiento, felicitación.

Un acto comunicativo debe transmitirse desde el hablante al oyente en forma de sentencia. Para ello utilizaremos un lenguaje de comunicación común. Preestablecido de mutuo acuerdo. Pero lo realmente importante en el acto comunicativo para el intercambio de conocimiento consiste en integrar una semántica en el lenguaje de comunicación.

#### **1.5.5.2.- Niveles y protocolos de comunicación**

Los protocolos de comunicación están normalmente especificados en diferentes niveles.

- *Nivel inferior:* especifica el método de interconexión
- *Nivel medio:* especifica el formato o sintaxis de la información que es transferida

- *Nivel superior*: especifica el significado o semántica de la información. La semántica no sólo se refiere a la esencia del mensaje, sino también al tipo de mensaje.

Existen protocolos de comunicación binarios y n-arios. Un *protocolo binario* involucra a un único emisor y a un único receptor, mientras que un protocolo *n-ario* involucra a un único emisor y a múltiples receptores (llamado “broadcasting” o “multicasting”). Un protocolo está especificado por una estructura de datos con los siguientes cinco campos.

1. Emisor
2. Receptor (o receptores)
3. Lenguaje en el protocolo
4. Funciones de codificación y decodificación
5. Acciones realizadas por el receptor o los receptores

Una decisión fundamental para la interacción de agentes es esperar la semántica del protocolo de comunicación (independiente del dominio) de la semántica del mensaje encapsulado (dependiente del dominio). El protocolo de comunicaciones debe ser conocido por todos los agentes. Debe ser conciso y tener un número limitado de actos de comunicación primitivos.

Lo visto hasta este momento sugiere la necesidad de un lenguaje que permita la intercomunicación entre nuestros agentes autónomos distribuidos. Existen principalmente dos propuestas que se están convirtiendo en especificaciones estándares que siguen la mayoría de sistemas de desarrollo de agentes. Una de ellas está basada en el lenguaje de comunicación **KQML** (*Knowledge Query and Manipulation Language*). Por otro lado hay otro grupo de investigadores que siguen las descripciones dadas por la organización **FIPA** (*Foundation for Intelligent Physical Agents*).

## 1.6.- ARQUITECTURA MULTIAGENTE FIPA

FIPA es una organización internacional dedicada a la promoción de la industria de los agentes inteligentes mediante el desarrollo de especificaciones que soporten la interoperabilidad entre agentes y aplicaciones basadas en agentes.<sup>22</sup>

Una de las implementaciones originalmente concebidas a partir de este estándar fue JADE (*Java Agent DEvelopment Framework*), el cual es una plataforma de gestión de sistemas multiagentes.

FIPA cubre todos los aspectos de un entorno de agentes como se presenta a continuación:

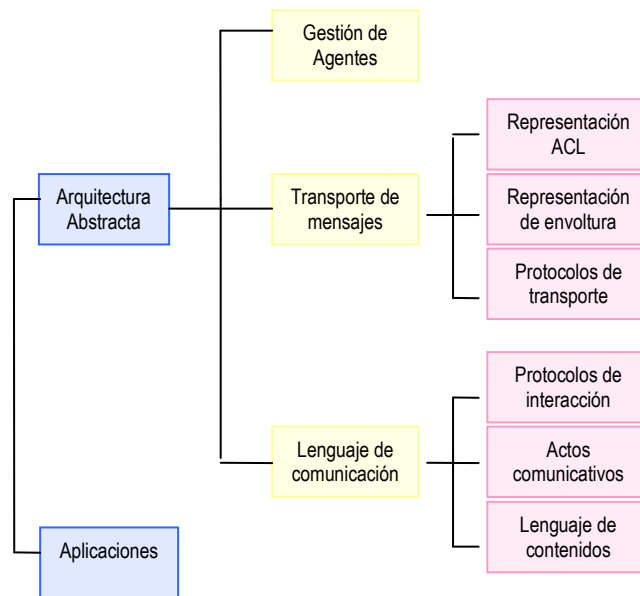


Figura 1.3 Arquitectura FIPA

La arquitectura de referencia FIPA se denomina Arquitectura Abstracta porque define los elementos comunes a diferentes sistemas subyacentes, los reúne en una especificación común e instancia esa arquitectura en forma de implementaciones diferentes pero interoperables.

<sup>22</sup> [www.fipa.org](http://www.fipa.org)

Su objetivo principal es conseguir un sistema totalmente abierto de tal forma que sistemas heterogéneos puedan interactuar a nivel de sociedades de agentes mediante el intercambio de mensajes con contenido semántico entre agentes, que pueden utilizar diferentes mecanismos de transporte, lenguajes de comunicación o lenguajes de contenido.

Esta arquitectura ofrece dos servicios de soporte a los agentes.

- Servicios de directorio (directory-services).
- Servicios de transporte de mensajes (message-transport services).

### **1.6.1.- SERVICIOS DE DIRECTORIO**

Proporciona un lugar en donde los agentes registran sus descripciones/servicios de tal forma que otros agentes puedan utilizar ese medio para localizar agentes/servicios con los que deseen interactuar/invocar.

Cada entrada de directorio se compone de:

- *Nombre del agente (AID)*: Global y único.
- *Localizador*: Este contiene un conjunto de *descripciones de transporte*. Es una estructura que contiene el tipo de transporte, una dirección específica para ese tipo y cero o más propiedades.
- *Atributos*. Son un conjunto de propiedades asociadas a un agente. Los atributos pueden utilizarse para llevar a cabo una búsqueda de agentes que cumplan ciertas características. En los atributos se puede recoger información sobre las ontologías que entiende el agente, servicios que proporciona, etc.

### **1.6.2.- GESTIÓN DE AGENTES**

El objetivo de la especificación de gestión de agentes es ofrecer un marco de trabajo estándar donde los agentes FIPA existan y operen, estableciendo un *modelo de referencia lógico* para la creación, registro, localización, comunicación, migración y baja de agentes.

### 1.6.2.1.- Plataforma FIPA

El modelo de referencia FIPA define la plataforma de agentes que se compone de un conjunto de entidades (Figura 1.4) que ofrecen diferentes servicios. A continuación se explican éstas entidades.

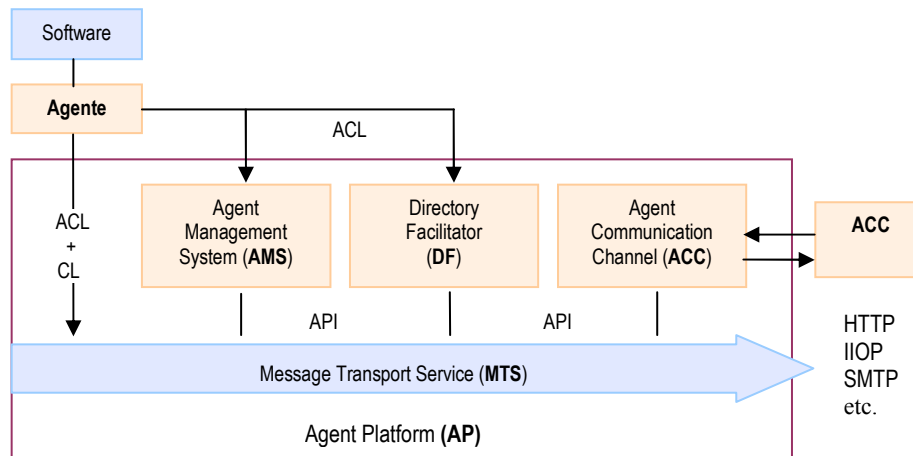


Figura 1.4 Plataforma FIPA

Entidad software (Software) sirve para representar cualquier sistema software accesible a través de un agente.

Plataforma de agentes (AP –Agent Platform) ofrece una infraestructura física sobre la que se despliegan los agentes y está compuesta de un software de soporte, los componentes para la gestión de agentes (DF, AMS, y MTS) y los agentes.

Agente Software (Agente) es un proceso computacional que implementa la funcionalidad autónoma y comunicativa de la aplicación, ofreciendo al menos un servicio. Los agentes se comunican a través de un ACL (*Agent Communication Language*). Un agente debe tener al menos un propietario y un identificador de agente (AID) que lo identifica de forma unívoca. Además, un agente puede disponer de varias direcciones de transporte a través de las cuales puede ser contactado. Dependiendo de la implementación de la plataforma un agente puede tratarse de un componente Java o un objeto CORBA.

Sistema de Gestión de Agentes (AMS –Agent Management System) Es el elemento de gestión principal, que conoce en todo momento el estado de la plataforma y de los agentes que pertenecen a ella. Entre sus responsabilidades (servicios que ofrece) está la creación, destrucción y control del cambio de estado de los agentes, supervisión de los permisos para que nuevos agentes se registren en la plataforma, control de la movilidad de los agentes, gestión de los recursos compartidos y gestión del canal de comunicación.

Cada AMS proporciona también un servicio de nombres (*Agent Name Service-ANS*), también llamado servicio de *páginas blancas*. Este servicio asocia el nombre que identifica a un agente en su sociedad y la dirección de transporte real en la que se encuentra. De tal forma que proporciona un método básico para buscar un agente dentro de la plataforma.

Como elemento de gestión, el AMS permite controlar el ciclo de vida de los agentes. Cada agente estará en un uno de los siguientes estados: Iniciado, Activo, Suspendido y Esperando.

Facilitador de Directorio (DF -Directory Facilitator) Como complemento al Servicio de Nombres, la plataforma de agentes incluye un servicio de *páginas amarillas*, que permite buscar un agente por sus capacidades y no sólo por su nombre. Esta labor la realiza el DF. Los agentes se registran en él indicando los servicios que ofrecen. Cuando otro agente tiene unas necesidades concretas lanza una búsqueda del servicio deseado obteniendo los agentes que le ofrecen estos servicios.

Canal de Comunicaciones (ACC -Agent Communication Channel). Todos los agentes FIPA deben tener acceso a un ACC. El ACC es el elemento encargado de gestionar el envío de mensajes entre agentes de una plataforma y entre agentes de distintas plataformas. Esta gestión consiste principalmente en el encaminamiento de los mensajes ACL desde su agente origen a su agente destino.

Cuando un agente en una plataforma quiere enviar un mensaje a otro en una localización remota, entrega este mensaje al ACC que se encargará de entregárselo al ACC de la plataforma donde se encuentre el otro agente. Este último ACC notificaría al agente

destino que ha recibido un mensaje. El modelo de comunicación entre agentes es asíncrono, lo que implica que: el ACC no se queda bloqueado ante el envío o recepción de mensajes, existen colas de envío y recepción de mensajes, para las cuales deben definirse políticas de gestión de colas de mensajes.

*Plataforma de Transporte de Mensajes (IPMT-APMT -Internal and Agent Platform Message Transport).* El IPMT (*Internal Platform Message Transport*) representa toda la infraestructura de comunicaciones que va a permitir que dos agentes dentro de la misma plataforma puedan comunicarse. El APMT (*Agent Platform Message Transport*) es otro elemento que hace la misma labor que el IPMT pero a nivel de comunicaciones entre plataformas (ACCs).

### **1.6.3.- TRANSPORTE DE MENSAJES**

El APMT (*Agent Platform Message Transport*) es el encargado de realizar la transferencia de mensajes ACL entre las ACCs de dos plataformas. Para describir este componente, se introduce la siguiente terminología:

*Servicio de Transporte de Mensajes (AMTS – Agent Message Transport Service)* ofrece un servicio de comunicación entre agentes, encargándose del transporte de mensajes entre agentes. Cada agente de la plataforma puede conseguir una referencia al AMTS a través de AMS (*Agent Management System*).

*Conversation Factory:* Es una interfaz implementada por el AMTS que permite a un agente establecer un contexto de comunicación (*Conversation Context*). Cuando un agente desea empezar una conversación con otro necesita obtener un nuevo contexto.

*Conversation Context:* Representa un extremo lógico de un canal de comunicación establecido para el intercambio de mensajes ACL. Define la información que necesita mantener el sistema para lograr un flujo de mensajes entre dos agentes. Un agente tendrá tantos objetos de este tipo como número de conversaciones en el que esté implicado. De forma muy básica se puede decir que un contexto de comunicación está definido por el

origen, el destino, un método de transporte y la calidad del servicio prestado. Además, cada objeto de esta clase contendrá una cola de mensajes en espera de ser recibidos por el agente y una función de *callback* mediante la que se notifica al agente de la llegada de nuevos mensajes pertenecientes a dicha conversación.

*Transport Factory*: Es una interfaz implementada por el AMTS que permite a un agente investigar el conjunto de mecanismos de transporte soportados (iiop, http, wap, etc.) y conocer algunas de las características de los mismos.

*Transport*: Representa un mecanismo de transporte de mensajes concreto. Cada vez que un agente desea establecer una nueva conversación tiene que identificar el método de transporte a usar o bien delegar esta responsabilidad al propio AMTS.

*Address*: Representa un extremo del canal de transporte utilizado. Dependiendo de del tipo de transporte elegido las direcciones serán diferentes (direcciones IP, URL, RFC\_822 email, etc.).

*Destination*: Identifica la entidad remota con la que se está produciendo la comunicación o intercambio de mensajes. Puede haber un solo destino asociado con varias direcciones correspondientes a diferentes tipos de transporte. Este modelo se basa en la vida real, en la que una persona puede tener varias direcciones de contacto tales como su teléfono personal, teléfono móvil, fax, dirección postal.

*Quality of Service*: Encapsula el conjunto de propiedades que gobiernan el modo en que el canal de transporte realizará el intercambio de mensajes entre los extremos de la comunicación. Algunas de estas propiedades son la integridad de datos, la privacidad, el rendimiento mínimo y el ancho de banda mínimo. Esto permitirá elegir en cada momento la calidad de servicio que se desea contratar de acuerdo con las necesidades.

*Message*: Es la unidad básica de información que se va transmitir por el canal de comunicación. Como ya se ha explicado, esta unidad va a ser el mensaje ACL.



### 1.6.3.1.- Estructura de un mensaje FIPA ACL

En la tabla 1.1 se puede observar la estructura de un mensaje FIPA-ACL, del cual los elementos performative, content, sender, receiver son obligatorios, el resto son opcionales

Ejemplo:

```
(request
  :sender (:name dominicagent@whitestein.com:8080)
  :receiver (:name rexhotel@tcp://hotelrex.com:6600)
  :ontology personaltravelassistant
  :language FIPASL
  :protocol fipa-request
  :content
    (action movenpickhotel@tcp://movenpick.com:6600
      (bookhotel (:arrival 25/11/2000) (:departure 05/08/2008) ...
    ))
)
```

<i>Elementos del mensaje FIPA-ACL</i>	
<i>Elemento</i>	<i>Descripción</i>
performative	Qué acción lleva a cabo el mensaje (CAL)
sender	Identidad del Emisor del mensaje
receiver	Identidad del Receptor del mensaje (un agente o una lista de agentes)
reply-to	Agente al que han de ser enviadas las respuestas (si no es el emisor)
content	Contenido del mensaje
lenguaje	El nombre del lenguaje de representación (CLL)
encoding	La codificación usada para el contenido
ontology	El nombre de la ontología utilizada
protocol	Identificador del protocolo de interacción que se está utilizando (IPL)
conversation-id	Identificador de una secuencia de actos comunicativos que forman parte de una misma conversación
reply-with	Responda con esta expresión ( Etiqueta para la respuesta)
in-reply-to	La etiqueta esperada en la respuesta
reply-by	Indicación del tiempo en el que se quiere que se responda al mensaje

**Tabla 1.1 Estructura de un mensaje FIPA-ACL**

### 1.6.3.2.- Representación de envoltorio

Cuando se envía un mensaje, se transforma utilizando una representación codificada apropiada para su transporte. El mensaje de transporte es el objeto transportado de agente a agente. Está constituido por el *payload* + *envelope*.

- *envelope*: contiene las descripciones de transporte del emisor y del receptor(es). Estas descripciones de transporte contienen información acerca de cómo se va a enviar el mensaje (tipo de protocolo, transporte, dirección). También puede contener algunos atributos adicionales (codificación utilizada, seguridad)
- *payload*: es el mensaje ya codificado en el formato adecuado para su inclusión en un mensaje de transporte. Aquí se encuentran los parámetros del mensaje y el contenido del mensaje.

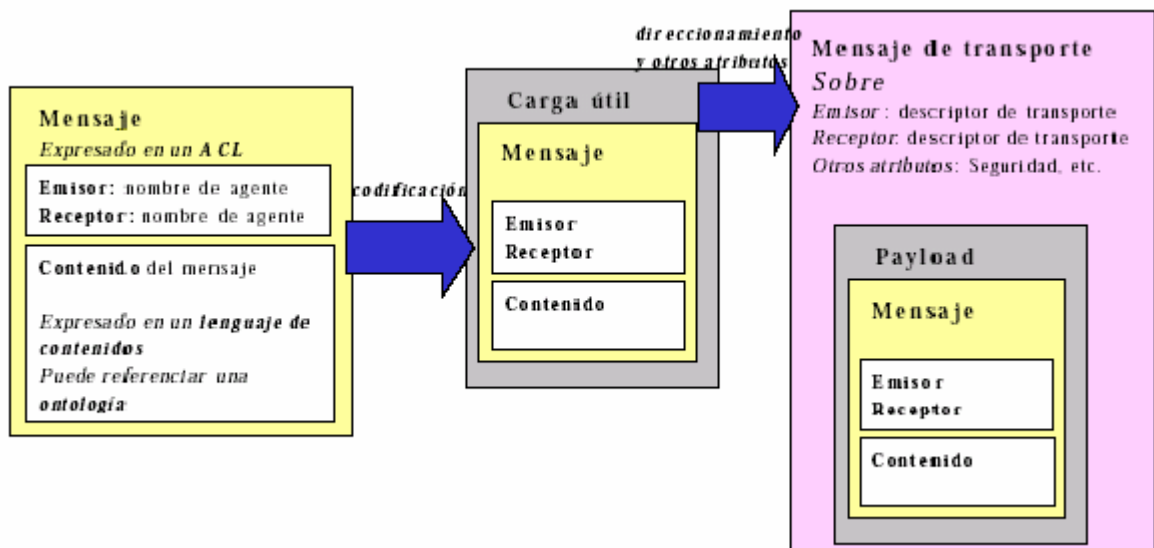


Figura 1.5 Envoltorio de un mensaje FIPA

#### 1.6.3.2.1.- Parámetros de envoltorio

to: A quien va dirigido el mensaje  
from: El que envía el mensaje  
Obligatorios  
acl-representation: representación ACL (String, XML, Bitefficient)  
date: fecha de la creación de la envoltura

- Opcionales*
- payload-length: Duración del byte de la carga útil
  - payload-encoding: lenguaje de codificación ACL(USASCII, UTF8)
  - received: Pone una evidencia de haber establecido el mensaje
  - security-object: certificación de encriptación e información

### 1.6.3.3.- Protocolos de transporte

Los protocolos de transporte de FIPA son:

- IIOP Internet Inter-ORB Protocol
- HTTP Hyper Text Transfer Protocol

### 1.6.4.- LENGUAJE DE COMUNICACIÓN FIPA-ACL

La comunicación entre agentes se basa en el paso de mensajes que representan actos de habla, y los agentes se comunican formulando mensajes escritos en un lenguaje de comunicación de agentes llamado FIPA ACL.

Un mensaje FIPA-ACL representa la intención de realizar alguna acción (acto comunicativo), es decir, cada mensaje puede ser visto como una acción del emisor, que trata de modificar el “estado mental” del receptor (y por lo tanto sus acciones). El estado de los agentes se describe con el lenguaje SL (*Semantic Language*).

Los mensajes FIPA-ACL están formados por:

- Lenguaje de Contenidos FIPA Content Language Library (CLL)
- Actos Comunicativos FIPA Communicative Act Library (CAL)
- Protocolos de Interacción FIPA Interaction Protocol Library (IPL)

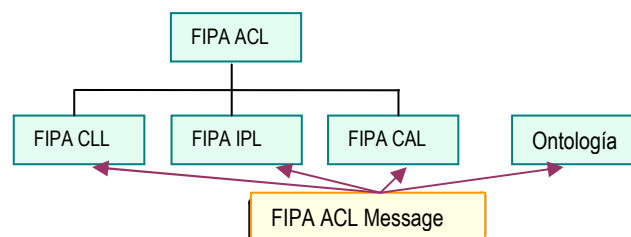


Figura 1.6 Componentes del Lenguaje de Comunicación FIPA-ACL

#### **1.6.4.1.- Lenguaje de contenidos**

Existen varios lenguajes usados como un lenguaje de contenidos. Ejemplo: KIF, Prolog, SQL, Serialized Objects, Binary Large Objects, FIPA-SL, FIPA-CCL, FIPA-RDF, FIPA-KIF. De las cuales, solamente la correspondiente al lenguaje SL tiene consideración de estándar.

##### **1.6.4.1.1.- Fipa-SL**

Es una lógica multimodal con operadores modales para creencias (B), deseos (D), creencias falsas (U), e intenciones u objetivos persistentes (PG).

Con SL se pueden representar:

- Propositiones
- Objetos
- Acciones

La semántica de cada acto comunicativo de FIPA ACL se define como un conjunto de fórmulas SL que describen:

- *Las precondiciones de factibilidad (FP)*: las condiciones necesarias para el emisor del acto comunicativo (aunque se den las condiciones el agente no está obligado a realizar el acto comunicativo, puede elegir).
- *El efecto racional (RE)*: lo que un agente espera que ocurra como resultado de realizar la acción (pero el agente receptor no está obligado a asegurar que ocurra el efecto esperado ya que incluso puede resultarle imposible lograrlo)

#### **1.6.4.2.- Actos Comunicativos**

Definen la semántica de los mensajes intercambiados y los diferentes lenguajes para expresar el contenido de un mensaje (lenguaje de contenido).

FIPA CAL es una librería compuesta por 22 acciones comunicativas que permiten representar las intenciones de la comunicación:

- accept-proposal    - agree    - cancel    - cfp    - confirm
- disconfirm        - failure    - inform    - inform-if    - inform-ref
- not-understood    - propagate    - propose    - proxy    - query-if
- query-ref         - refuse    - reject-proposal    - request    - subscribe
- request-when     - request-whenever

A continuación se presentan los actos comunicativos agrupados por sus funciones:

Nombre	Paso de Información	Solicitud de información	Negociación	Ejecución de acciones	Manejo de errores
accept-proposal					
agree					
cancel					
cfp					
confirm					
disconfirm					
failure					
inform					
inform-if					
inform-ref					
not-understood					
propose					
query-if					
query-ref					
refuse					
reject-proposal					
request					
request-when					
request-whenever					
subscribe					

Tabla 1.2 Actos comunicativos de FIPA

### 1.6.4.3.- Protocolos de Interacción

Para poder establecer una conversación entre agentes es necesario definir previamente el protocolo de interacción (llamado conversación) que van a seguir durante la conversación. Un agente puede participar simultáneamente en múltiples diálogos con diferentes agentes y con diferentes protocolos de interacción. Para utilizar un protocolo en una conversación, los agentes deben escribir el nombre del protocolo a utilizar en el parámetro: *protocol*.

- Un protocolo termina cuando se alcanza el último mensaje del protocolo o se elimina el nombre del protocolo del parámetro: *protocol*
- Cuando un agente no conoce un determinado protocolo tiene que devolver un mensaje *refuse* explicando el motivo por el que rechaza la comunicación
- Si durante el seguimiento de un protocolo, un agente recibe un mensaje no contemplado en el mismo tiene que devolver un mensaje *not-understood*. Está prohibido responder un *not-understood* con otro *not-understood* para no caer en bucles infinitos.

FIPA IPL es una librería compuesta por 11 protocolos de interacción:

- |                              |                        |                     |
|------------------------------|------------------------|---------------------|
| - FIPA-propose               | - FIPA-query           | - FIPA-subscribe    |
| - FIPA-request               | - FIPA-request-when    | - FIPA-contract-net |
| - FIPA-iterated-contract-net | - FIPA-auction-english | - FIPA-recruiting   |
| - FIPA-auction-dutch         | - FIPA-brokering       |                     |

### 1.6.5.- SERVICIO DE ONTOLOGÍA

Un mensaje ACL consiste básicamente en una expresión de un determinado lenguaje y con un conjunto de términos específicos de la ontología usada. Esto asegura que estos agentes atribuyen el mismo significado a los símbolos utilizados en los mensajes.

Una ontología es un conjunto de símbolos (constantes) asociados con su correspondiente interpretación o significado. Cuando un agente se registra en el DF

(*Directory Facilitator*) informa de las ontologías de las que tiene conocimiento, de tal forma que, cuando otro agente quiere conversar con él utilizará una de ellas a fin de conseguir un entendimiento mutuo.

En un dominio pequeño es posible que las ontologías aplicables se incluyan en el propio código del agente. Cuando pensamos en sistemas de mayor envergadura se debe manejar Servicios de Ontología, de tal forma que todos los agentes puedan utilizar dichos servicios para obtener toda la información necesaria.

Las labores básicas del Servicio de Ontología son:

- Mantiene un conjunto de ontologías de uso público accesible a los agentes.
- Traduce expresiones entre diferentes ontologías.
- Responde a consultas sobre términos de las ontologías que gestiona.
- Facilita la identificación y uso de ontologías compartidas entre los agentes.
- Descubre nuevas ontologías y las pone a disposición de todos los agentes.

En la actualidad existen una serie de servidores de ontologías fuera del ámbito de FIPA, tales como el servidor de Ontolingua, el servidor de XML/RDF y el servidor ODL a los que se accede a través de APIs (*Application Programming Interface*) como OKBC, HTTP y ODL respectivamente. FIPA no pretende sustituir estas implementaciones, sino agruparlas de forma que se puedan utilizar de forma transparente dentro de la plataforma.

#### **1.6.6.- SEGURIDAD DE AGENTES**

Las recomendaciones de FIPA están basadas en la utilización de los estándares de seguridad X.509, basada en infraestructuras de seguridad de clave pública (PKI).

Este estándar presenta un modelo de seguridad asimétrico o basado en certificados de clave pública y clave privada. Para cada entidad se definen dos tipos de claves, la clave privada a las que solo tiene acceso la entidad propietaria del certificado, y la clave pública a la que tienen acceso todas las demás entidades de la sociedad. Por cada entidad, la autoridad certificadora correspondiente genera un par de claves (pública y privada) para el

cifrado (confidencialidad) y otro par de claves (pública y privada) para la firma (autenticidad). Cada entidad recibe sus claves privadas por parte de la autoridad certificadora. Las claves públicas de todas las entidades están guardadas en una estructura de directorio (LDAP) y pueden ser consultadas en cualquier momento. También es posible que una entidad distribuya sus claves públicas a las entidades con las que quiere comunicarse de forma segura.

Son posibles otros modelos de seguridad de más bajo nivel como el uso de sockets seguros, SSL (*Secure Socket Layer*), a nivel de la capa de transporte.

### **1.6.7.- MOVILIDAD DE AGENTES EN LA PLATAFORMA**

Siguiendo la filosofía del estándar, FIPA no propone soluciones tecnológicas sino más bien aporta ideas acerca de cómo integrar de forma homogénea y natural las capacidades de los agentes móviles dentro de la plataforma. La interoperabilidad entre sistemas es muy importante en este campo, pues si dos sistemas de distinto origen no pudiesen intercambiar agentes la movilidad no sería aprovechable.

Cuando se trata de agentes móviles podemos trabajar con dos soluciones, más o menos complejas de llevar a cabo:

*Migración:* Este modelo es más complejo, pues requiere la transferencia de código y datos del agente a la plataforma remota. Cuando un agente va a migrar se "suspende", se transfiere por la red y se "despierta" en el otro extremo prosiguiendo la ejecución de sus tareas en el mismo punto donde las había dejado.

*Clonación:* En este caso se crea una copia del agente en el nodo remoto que se encarga de hacer el trabajo solicitado por su clon en la máquina origen. En este caso no hay una transferencia efectiva de código y datos, sino, sólo un conjunto de órdenes que el clon debe realizar.



FIPA define dos modelos de movilidad en función de quién tenga que hacer la gestión de la transferencia:

*Modelo Simple:* El AMS (*Agent Management System*) es el responsable de realizar toda la gestión necesaria. El agente solicita a su AMS la transferencia y éste se ocupa de llevarla a cabo.

*Modelo Complejo:* El agente es el responsable de realizar todas las operaciones que le permiten desplazarse a la plataforma remota.

### 1.6.8.- NOMBRADO DE AGENTES (AGENT NAME SERVICE-ANS)

En la actualidad el modelo de nombrado de agentes de FIPA es más flexible y completa para acometer futuras necesidades.

```
AgentName=      (":guid" Word NameFields*)
NameFields=    "addresses" ("CommAddress+ ")
                | ":resolver" ("AgetName+ ")
                | ":authenticator" ("AgentName+ ")
```

- *GUID:* Identificador único del agente en la plataforma de agentes. No tiene porque contener la dirección de transporte como en el modelo simple.
- *Addresses:* Dirección/es de transporte en la que se encuentra el agente.
- *Resolver:* Dirección donde se encuentra el Servidor de Nombres (ahora en el AMS)
- *Authenticator:* Dirección del agente de autenticación (ahora el propio AMS).

Un ejemplo podría ser:

```
:guid          MiAgente@iiop://www.ucm.es
:resolver (    :guid          ams@www.ucm.es
               :addresses    (iiop://www.ucm.es:81/ams
                               http://www.ucm.es:80/ams) )
:addresses     (iiop://fipa.org/MiAgente)
:authenticator (iiop://fipa.org/Autenticación)
```

## CAPÍTULO II

### II.- METODOLOGÍA INGENIAS

#### 2.1.- INTRODUCCIÓN

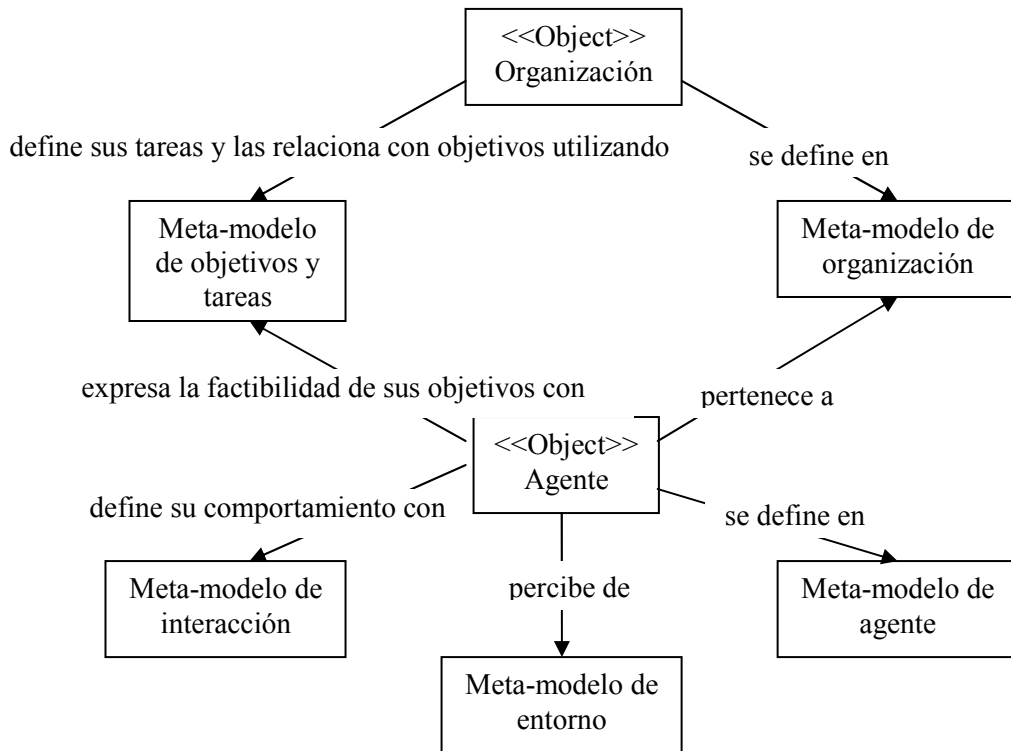
En el desarrollo de Sistemas Multiagentes (SMA) se estudia al SMA como un sistema software que hay que construir. El desarrollo no parte de cero, sino que utiliza plataformas de desarrollo de agentes que proporcionan servicios básicos de comunicación, gestión de agentes y una arquitectura de agente. Para desarrollar este tipo de sistemas se necesitan metodologías que estructuren el desarrollo de acuerdo con las prácticas de ingeniería del software.

Para ello existe la metodología INGENIAS<sup>23</sup>, que concibe el SMA como la representación computacional de un conjunto de modelos. Cada uno de estos modelos muestra una visión parcial del SMA: los agentes que lo componen, las interacciones que existen entre ellos, cómo se organizan para proporcionar la funcionalidad del sistema, qué información es relevante en el dominio y cómo es el entorno en el que se ubica el sistema a desarrollar.

Para especificar cómo tienen que ser estos modelos se definen meta-modelos. Un meta-modelo es una representación de los tipos de entidades que pueden existir en un modelo, sus relaciones y restricciones de aplicación.<sup>24</sup> Los meta-modelos que se describen en esta metodología son una evolución del trabajo realizado en la metodología MESSAGE. Ahora el resultado de INGENIAS, son cinco meta-modelos que giran alrededor de dos entidades la *organización* y el *agente*. Figura 2.1

---

<sup>23</sup> <sup>24</sup> Web Oficial Metodología <http://grasia.fdi.ucm.es/ingenias/Spain/lenguaje/index.php>



**Figura 2.1 Relaciones entre los diferentes meta-modelos y las dos entidades principales, la organización y el agente.**

El meta-modelo de agente describe agentes particulares y los estados mentales en que se encontrarán a lo largo de su vida. El meta-modelo de interacción se ocupa de detallar cómo se coordinan y comunican los agentes. El meta-modelo de tareas y objetivos se usa para asociar el estado mental del agente con las tareas que ejecuta. El meta-modelo de organización define cómo se agrupan los agentes, la funcionalidad del sistema y qué restricciones hay que imponer sobre el comportamiento de los agentes. Por último, el modelo del entorno define qué existe alrededor del nuevo sistema.<sup>25</sup>

En resumen, la línea que sigue INGENIAS es la de una ingeniería del software orientada a agentes según las ideas de Pressman,<sup>26</sup> quien concibe tres elementos en una ingeniería del software: herramientas, métodos y procedimientos. La metodología proporciona meta-modelos para definir el SMA, herramientas de soporte para generarlos y

<sup>25</sup> Web Oficial Metodología <http://grasia.fdi.ucm.es/ingenias/Spain/lenguaje/index.php>

<sup>26</sup> Pressman, R. S.: Software Engineering: A Practitioner's Approach. Libro completo. McGraw-Hill Series in Software Engineering and Technology. McGraw-Hill, Inc. 1982

hacerlos progresar en las distintas etapas del ciclo de vida (análisis, diseño e implementación) y un conjunto de actividades que estructuran el desarrollo del SMA.

## **2.2.- META – MODELADO**

Esta metodología utiliza meta-modelos para definir cómo debe ser la especificación del sistema y herramientas de soporte para facilitar el proceso de generación de especificaciones, por lo tanto los meta-modelos son especificaciones para la construcción de modelos generadas con lenguajes de meta-modelado. Un meta-modelo define las primitivas y las propiedades sintácticas y semánticas de un modelo.<sup>27</sup>

La ventaja de utilizar meta-modelos es que las especificaciones generadas de los SMA son lo suficientemente estructuradas para ser procesadas de forma automática. Así, se puede plantear la verificación automática de la construcción de los modelos para ver si cumplen restricciones identificada por el desarrollador, generar documentación del sistema en diferentes formatos de diferentes partes de los modelos, e incluso establecer la generación automática de código desde la información recogida en los modelos.

Los lenguajes de meta-modelado considerados son dos: *Meta-Object Facilities* (MOF)<sup>28</sup> y *Graph, Object, Property, Relationship, and Role* (GOPRR)<sup>29</sup>. Ambos son similares y de hecho se pueden hacer traducciones de uno a otro. En esta metodología la definición de los meta-modelos se hace utilizando notación UML siguiendo las restricciones de GOPRR.

### **2.2.1.- NOMENCLATURA**

En esta metodología se ha establecido una nomenclatura de relaciones, las cuales siguen unas reglas nemotécnicas que ayudan a identificar de qué meta-modelo o parte

---

<sup>27</sup> Web Oficial Metodología <http://grasia.fdi.ucm.es/ingenias/Spain/faq.php>

<sup>28</sup> OMG: MOF. Meta Object Facility (specification). Informe. 3-4-2000b

<sup>29</sup> Lyytinen, K. S. y Rossi, M.: METAEDIT+ --- A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. Actas de conferencia. Springer-Verlag. LGNS#1080.1999

específica del meta-modelo han surgido. El nombre de la relación es precedido por un conjunto de letras que denota su procedencia, como el flujo de trabajo (*WF*), meta-modelo de agente (*A*), interacción (*I*), unidad de interacción (*UI*), modelos de tareas y objetivos (*GT*), relaciones sociales (*AGO*), organización (*O*) o el entorno (*E*). Esta nomenclatura es utilizada al modelar el SMA con la herramienta IDK (Ingenias Development Kit)<sup>30</sup>.

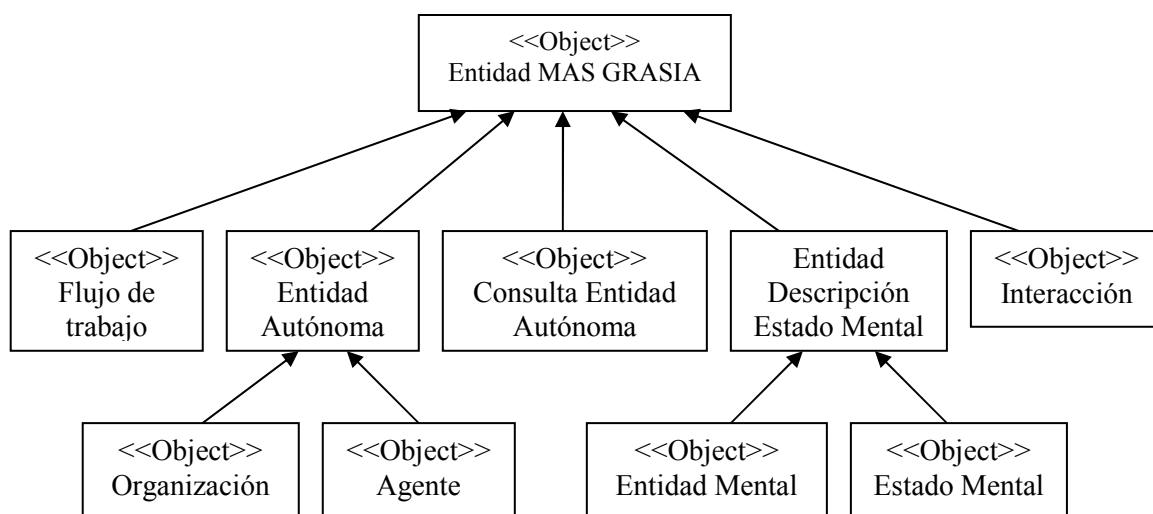
<p>IdentificadorRelacion ::= RelacionFlujoTrabajo   RelacionAgente    RelacionInteraccion   RelacionUnidadInteraccion    RelacionMetaTarea   RelacionSocial    RelacionOrganización   RelacionEntorno  </p> <p>RelacionFlujoTrabajo ::= <b>WF</b> Identificador  RelacionAgente ::= <b>A</b> Identificador  RelacionInteraccion ::= <b>I</b> Identificador  RelacionUnidadInteraccion ::= <b>UI</b> Identificador  RelacionMetaTarea ::= <b>GT</b> Relacion  RelacionSocial ::= <b>AGO</b> Relacion  RelacionOrganización ::= <b>O</b> Relacion  RelacionEntorno ::= <b>E</b> Relacion</p>
--

**Figura 2.2 Nomenclatura para los nombres de las relaciones**

### **2.2.2.- ENTIDADES BÁSICAS**

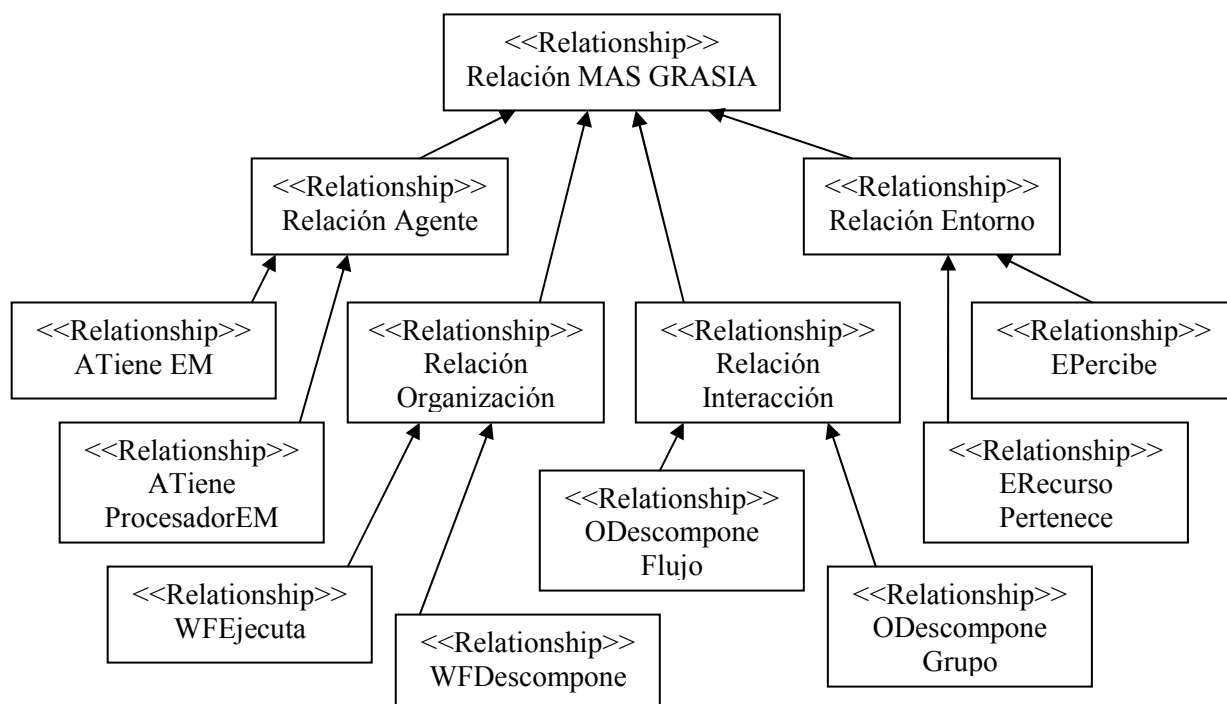
La metodología proporciona una jerarquía de conceptos básicos para el desarrollo de un SMA así como una notación para representar estos conceptos. La jerarquía comienza con la Entidad MAS GRASIA y la Relación MAS GRASIA, que reciben este nombre por el grupo de investigación en que han sido desarrolladas (GRupo de Agentes Software del departamento de sistemas Informáticos y progrAmación o GRASIA).<sup>31</sup>

<sup>30</sup> <sup>31</sup> Tesis Doctoral de INGENIAS: <http://www.fdi.ucm.es/profesor/jpavon/doctorado/practicas/index.html>



**Figura 2.3 Entidades básicas de la metodología**






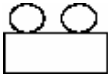
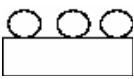

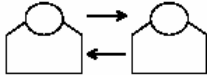
Los meta-modelos que se van a presentar se fundamentan también en la existencia de relaciones contextualizadas en cuatro dominios: agentes aislados, organizaciones de agentes, interacciones entre agentes y el entorno de los agentes. Estas relaciones asocian especializaciones de la Entidad MAS GRASIA. Se trata de relaciones dirigidas que parten de una entidad para ir a un conjunto finito de entidades destino.



**Figura 2.4 Relaciones básicas de la metodología**

### 2.2.3.- NOTACIÓN

A continuación se presenta la notación utilizada en INGENIAS:<sup>32</sup>

	<b>Objetivo.</b> Se etiqueta con el nombre del objetivo
	<b>Rol.</b> Se etiqueta con el nombre del rol.
	<b>Procesador de estado mental.</b> Se etiqueta con el nombre del procesador.
	<b>Gestor de estado mental.</b> Se etiqueta con el nombre del gestor.
	<b>Agente.</b> Se etiqueta con el nombre del agente.
	<b>Grupo.</b> Se etiqueta con el nombre del grupo.
	<b>Organización.</b> Se etiqueta con el nombre de la organización.
	<b>Flujo de trabajo.</b> Se etiqueta con el nombre del flujo.
	<b>Interacción.</b> Se etiqueta con el nombre de la interacción y su naturaleza, como coordinación, planificación o negociación.

<sup>32</sup> Tesis Doctoral de INGENIAS: <http://www.fdi.ucm.es/profesor/jpavon/doctorado/practicas/index.html>




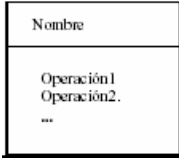
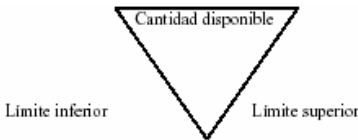
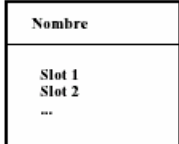

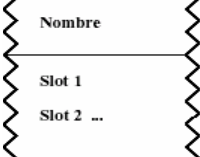
	<p><b>Consulta de entidades autónomas.</b> Se etiqueta con nombres concretos de agentes existentes o expresiones que denotan agentes existentes.</p>
	<p><b>Unidad de interacción.</b> Se etiqueta con el nombre de la unidad y el acto del habla al que hace referencia, como request, inform, o not-understood.</p>
	<p><b>Tarea.</b> Se etiqueta con el nombre de la tarea.</p>
	<p><b>Aplicación.</b> Se etiqueta con el nombre de la aplicación y las operaciones soportadas.</p>
	<p><b>Recurso.</b> Se etiqueta con el nombre del recurso, la cantidad disponible del mismo, el límite inferior y superior admisible. Por debajo o encima de estos límites, el recurso se deshabilita.</p>
<p><b>HECHO</b></p> 	<p><b>Hecho.</b> Se etiqueta con el nombre del hecho y los nombres de los slots identificados.</p>
	<p><b>Creencia.</b> Se etiqueta con el nombre de la evidencia e información acerca de qué es lo que se está aceptando como cierto.</p>
	<p><b>Evento.</b> Se etiqueta con el nombre del evento y los nombres de los slots identificados.</p>

Tabla 2.1 Notación empleada en la representación de los modelos



## 2.3.- META - MODELO DE AGENTE

Este meta-modelo se centra en la funcionalidad del agente y en el diseño de su control. Describe al agente individualmente excluyéndolo de las interacciones con otros agentes ya que el agente (*entidad autónoma*) se caracteriza por tener propósitos y una identidad única. En resumen este meta-modelo proporciona información acerca de los siguientes aspectos:<sup>33</sup>

- Responsabilidades: Se trata de las tareas que sabe ejecutar y de los objetivos que se compromete a alcanzar.
- Comportamiento: Aquí se habla del control del agente, esto es, mediante qué mecanismos se va a asegurar la ejecución de tareas dentro de los parámetros acordados.

Existe una variedad de modelos de comportamiento (control, estado mental y evolución del estado mental) y definición de responsabilidades de agentes (asignación de tareas y declaración de habilidades). La Tabla 2.2 resume las características que presenta la plataforma de desarrollo de SMA JADE y en la cual se puede observar que sigue un enfoque más de ingeniería, centrando su atención en las tareas y dejando el estado mental como un complemento.

Para explicar mejor cómo es el meta-modelo de agente, se lo ha estructurado en tres partes: Control de Agentes, Asociación de Responsabilidades y Definición del Estado Mental.

---

<sup>33</sup> Web Oficial Metodología: <http://grasia.fdi.ucm.es/ingenias/Spain/lenguaje/magente.htm>

Plataforma	Estado mental	Evolución estado mental	Control	Responsabilidades
JADE	No existe	Codificado por el usuario dentro de las clases de comportamiento. Se permiten añadir/quitar comportamientos	Ejecución de las instancias de clases de comportamiento, siguiendo una política <i>round-robin</i> no expropiativa entre aquellos comportamientos concurrentes	Asignación de tareas directa. Las tareas se codifican dentro de las clases de comportamiento

Tabla 2.2 Solución de JADE para el modelado del comportamiento y responsabilidades

### 2.3.1.- CONTROL DE AGENTE

El control de agente se basa en objetivos y tareas. El agente juega cierto rol, el cual le lleva a perseguir unos objetivos. Estos objetivos se alcanzan cuando el mundo, tal como lo concibe el agente, se ha modificado. Las tareas son las encargadas de producir esta modificación. La modificación se expresa mediante la producción de evidencias, que pueden ser nuevas entidades mentales o eventos disparados por el entorno. Este esquema se incluye en el meta-modelo a través del estado mental, el gestor de estado mental y el procesador de estado mental.

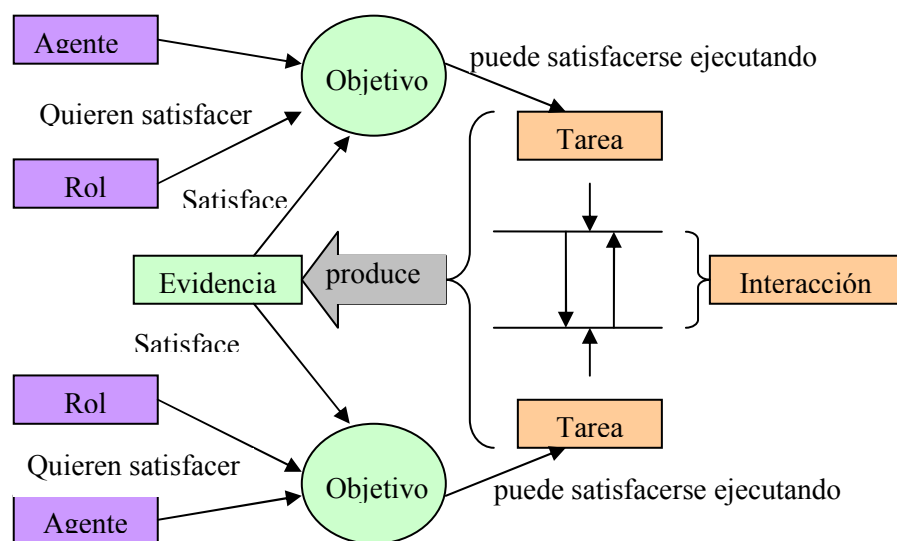


Figura 2.5 Conceptos relevantes en el control del agente

El estado mental puede verse como toda aquella información que permite al agente tomar decisiones. El propósito del Gestor del Estado Mental es desarrollar la evolución del estado mental mediante las operaciones de creación, destrucción, modificación y monitorización del conocimiento del agente. Es responsable de mantener la coherencia del conocimiento almacenado y de hacerlo evolucionar. El propósito del Procesador del Estado Mental es la toma de decisiones (el control del agente). Las decisiones se pueden producir de forma algorítmica, en función de las tareas a ejecutar y el beneficio a obtener, u obedecer a la consecución de objetivos eligiendo secuencia de tareas a ejecutar.

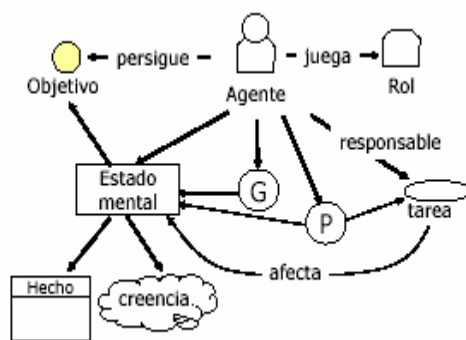


Figura 2.6 Elementos del modelo de Agente

Para definir los requisitos que deben satisfacer el gestor y el procesador se utilizan tres herramientas: relaciones entre las tareas y entidades mentales, relaciones entre el agente y objetivos, y especificación de estados intermedios por los que pasa un agente en ejecución.

### 2.3.1.1.- Agentes en Ejecución

Para describir la evolución del estado mental, es necesario, en este meta-modelo hacer referencia a propiedades del agente en ejecución. Para ello se tiene la entidad Consulta Entidad Autónoma que sirve para describir quién participa en las interacciones, jugando los roles correspondientes. El símbolo de Consulta Entidad Autónoma se representa con un agente encerrado entre paréntesis y se etiqueta de diferente manera dependiendo de a qué se haga referencia:

- Un agente concreto (Agente Concreto).- En este caso la etiqueta es el nombre del agente o bien una expresión que denota un único agente. Como caso especial, se tiene la etiqueta “(ejecutor)”. Aparece al referenciar un agente que va a ejecutar una tarea, al definir un estado mental intermedio de un agente en una interacción, o al estudiar la satisfacción de un objetivo.

- *Un conjunto de agentes (Consulta Requisitos Agente).*- Otra vez la etiqueta define un conjunto de agentes por intensión. Se utiliza una expresión que denota las propiedades que se requieren de los agentes. Estas expresiones utilizan como términos predicados equivalentes a las relaciones representadas en los meta-modelos.

### **2.3.2.- ASOCIACIÓN DE RESPONSABILIDADES**

El meta-modelo contempla el uso de roles para asignar responsabilidades. En general, el *rol* es una abstracción de un conjunto de funciones, que puede tener estado y que para existir necesita de otra entidad no abstracta que lo desempeñe, en este caso el agente. De esta forma es posible aplicar los roles para describir conjuntos de responsabilidades (las funciones asociadas), protocolos (gracias a que el rol tiene estado implícito), o permisos de actuación (cambiando los roles de un agente, se cambia su capacidad de actuación).

Los roles aparecen en las interacciones y en las organizaciones. Los roles carecen de gestores y procesadores de estado mental, y, por lo tanto, de capacidad de toma de decisiones. Los roles también fomentan la reusabilidad, ya que existen para otorgar al que los juega sus cualidades. Esta meta-relación indica al diseñador que el agente adquiere todos los objetivos asociados al rol, todas sus responsabilidades y sus capacidades.

### **2.3.3.- ESTADO MENTAL**

El estado mental se define como agregación de entidades mentales: objetivos, creencias, compromisos. Adicionalmente, se plantean hechos (información cierta para el agente) y eventos (los cambios ocurridos, en el mundo que el agente capta). Los eventos deberían transformarse en hechos automáticamente en cuanto formaran parte del estado mental del agente, sin embargo, en este modelo conviene más el mantenerlos así y dejar que sean los procesos internos del agente los que decidan si debe realizarse tal transformación o simplemente deshacerse del evento.

Se admiten múltiples instancias del estado mental relacionadas con un agente. Al asociarlo directamente, se indica que el agente, al crearse, parte de ese estado. Cuando se asocia indirectamente, se está definiendo un estado mental intermedio en la vida del agente.

## 2.4.- META – MODELO DE INTERACCIÓN

Las interacciones determinan el comportamiento de los agentes cuando actúan sobre ellos. El comportamiento es en función de los objetivos de los agentes y las tareas a ejecutar. El presente meta-modelo se construye sobre: agentes, roles, objetivos, interacciones y unidades de interacción. Los agentes y roles son los actores de las interacciones. En las interacciones se ejecutan unidades de interacción (pasos de mensaje, lectura y escritura en un espacio de tuplas) en las que hay un iniciador (emisor) y colaboradores (receptores). Además, se justifica la participación de los actores en la interacción y la existencia de la interacción en sí, mediante objetivos.<sup>34</sup>

Estos elementos se interrelacionan mediante el conjunto de asociaciones, las cuales se categorizan en cuatro grupos: naturaleza, contexto, ejecución y notación. (Figura 2.7)

El *contexto* ayuda al ingeniero a situar la interacción en el marco del SMA, estableciendo en qué condiciones se iniciarán las interacciones. Esta información se utiliza para codificar el comportamiento del agente en forma de reglas de producción o para generar métodos de validación de la interacción, como métodos de detección de interbloqueos.

La *naturaleza* de la interacción indica qué clase de interacción se está considerando con respecto al tipo de control. Esta clase de control es de lo que trata la coordinación. La naturaleza de la interacción es importante porque determina qué algoritmos, herramientas y guías pueden ayudar al diseñador. El tipo de interacción también muestra qué elementos tienen que considerarse. Por ejemplo, si la naturaleza es competitiva entonces debería existir en la interacción una referencia al objeto por cuya posesión o uso se compete.

---

<sup>34</sup> Web Oficial Metodología <http://grasia.fdi.ucm.es/ingenias/Spain/lenguaje/minteraccion.htm>

La *ejecución* de la interacción se refiere al conjunto de actividades requeridas a la hora de desarrollar la interacción. En la ejecución existe un orden impuesto sobre las acciones y unidades de interacción. El orden establece el protocolo de la interacción, mientras que las unidades representan mensajes o protocolos, dependiendo del nivel de abstracción.

La *representación* toma la información de la ejecución y la naturaleza para crear una representación gráfica o formalismo textual sencillo de manejar por los ingenieros. Por ejemplo, una negociación que siga contract-net puede expresarse con diagramas de protocolo de Agent UML, diagrama de protocolos FIPA o varios diagramas de secuencia UML.

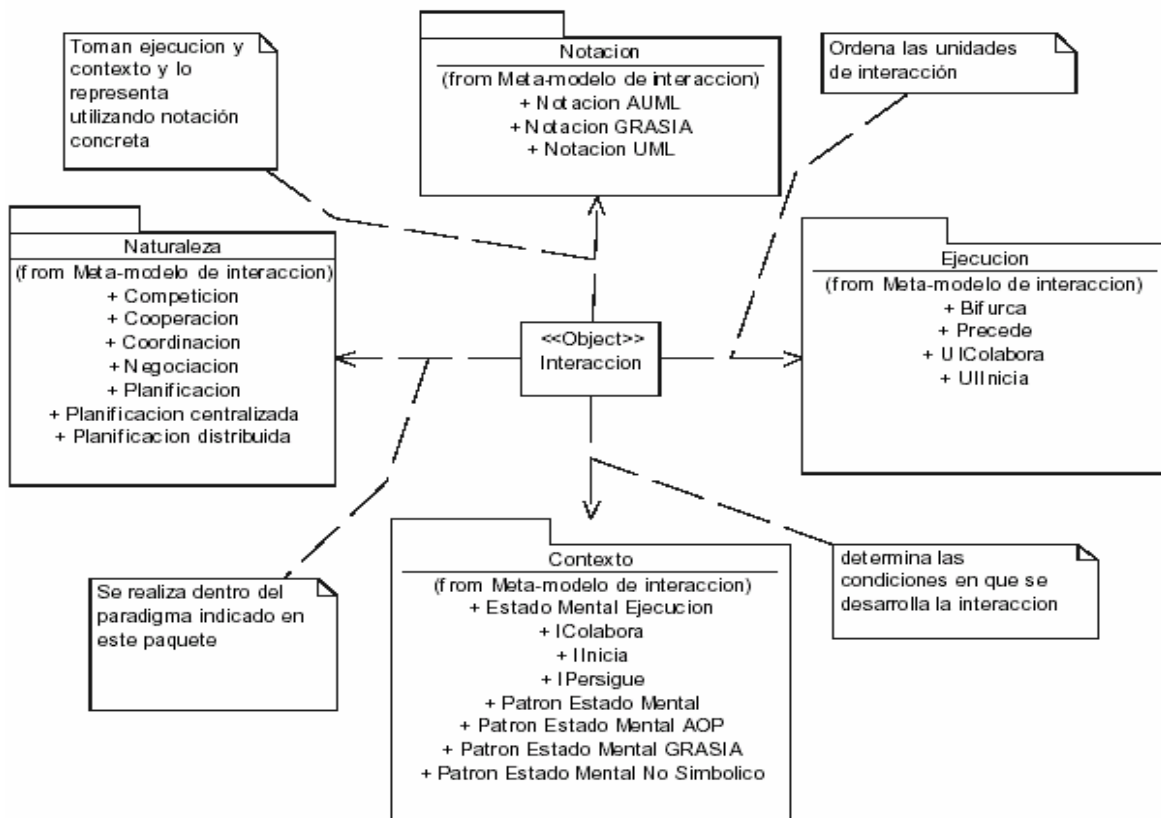


Figura 2.7 Relación entre los diferentes aspectos de la interacción

En el área de agentes, el análisis puede ejecutarse de una forma similar como en el Proceso Racional Unificado (*Rational Unified Process*),<sup>35</sup> esto es, generar casos de uso como primer paso, diagramas de colaboración, secuencia o de actividades para mostrar cómo evoluciona el sistema. Pero en el diseño no puede ser así ya que el modelado de las interacciones se debe definir a alto nivel con diagramas de paso de mensajes *Message Sequence Charts*,<sup>36</sup> diagramas de secuencia y colaboración (UML) y de protocolos (FIPA y AUML).

## 2.5.- META – MODELO DE OBJETIVOS Y TAREAS

El propósito del presente meta-modelo es de recoger las motivaciones del SMA, definir las acciones identificadas en los modelos de organización, interacciones o de agentes y cómo afectan estas acciones a sus responsables. Esta información constituye parte de la especificación de cómo se quiere que sea el control del agente a alto nivel. La justificación del uso de objetivos como representación del control del agente, se encuentra en el principio de racionalidad, en donde las acciones del agente se justifican por los objetivos que persigue, lo cual lleva también a la asociación de tareas y objetivos (Figura 2.8).<sup>37</sup>

Finalmente, la inclusión de las tareas en este meta-modelo obedece a que juegan un papel fundamental en la evolución del estado mental de sus responsables.

En esta metodología una *tarea* se puede ver como transformadora del estado global, el cual concibe la tarea como pre-condiciones y post-condiciones, esto permite su incorporación en mecanismos de planificación y razonamiento y tarea como proceso, donde la tarea será un conjunto de instrucciones que han de ejecutarse. En concreto, la tarea es un proceso integrado en un flujo de trabajo, planteamiento similar al que se adopta en el meta-modelo de organización.

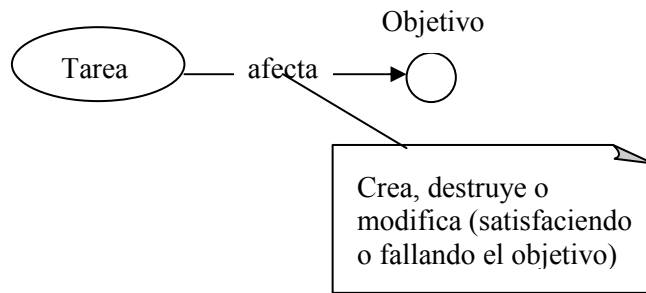
---

<sup>35</sup> Jacobson, I., Rumbaugh, J. y Booch, G.: The Unified Software Development Process. Libro completo. Addison-Wesley. 1999.

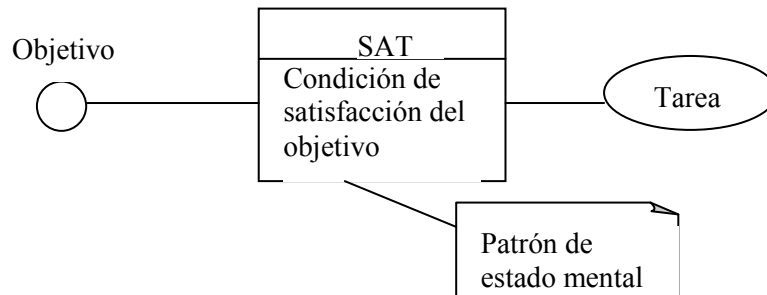
<sup>36</sup> International Telecommunication Union: ITU-120: Formal Description Techniques (FDT): Message Sequence Chart. Informe. 99a

<sup>37</sup> Web Oficial Metodología <http://grasia.fdi.ucm.es/ingenias/Spain/lenguaje/mtareas.htm>

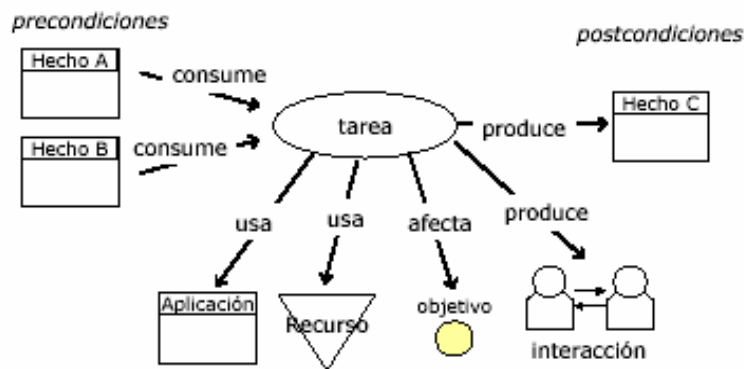
- Una tarea afecta a un objetivo



- La satisfacción de un objetivo justifica la elección de una tarea



**Figura 2.8 Relaciones entre tareas y objetivos**



**Figura 2.9 Elementos de definición de Tareas**

Los *objetivos* son las entidades de control, porque indican al agente qué es lo que le motiva, y el compromiso, porque liga al agente con una acción requerida por otro actor. Los objetivos tienen asociado un estado mediante la propiedad estados posibles, que es una lista de constantes que denotan los estados por los que pasa. Los objetivos se satisfacen con la presencia de ciertas evidencias y pueden darse por fracasados cuando estas evidencias



no se han producido. Hasta llegar a un objetivo satisfecho o fracasado, se pueden recorrer varios caminos. Aquí se propone que un objetivo pueda estar pendiente, recogiendo evidencias, refinado, en proceso, satisfecho o fallido. El ciclo de vida hace referencia a cómo se ejecutan tareas asociadas al objetivo (las que producen las evidencias que necesita). Las tareas no tienen estado en esta propuesta, por lo que, en caso de necesitar estructurar su ejecución, las tareas tienen que delegar su estado a otra entidad, el objetivo en este caso. La elección de este ciclo de vida no es obligatoria.

## **2.6.- META – MODELO DE ORGANIZACIÓN**

El modelo de organización es el equivalente a la arquitectura del sistema en un SMA. Para simplificar la presentación del presente meta-modelo, se lo ha dividido en tres partes:<sup>38</sup>

- Descripción estructural de la organización
- Descripción funcional
- Descripción social

### **2.6.1.- DESCRIPCIÓN ESTRUCTURAL DE LA ORGANIZACIÓN**

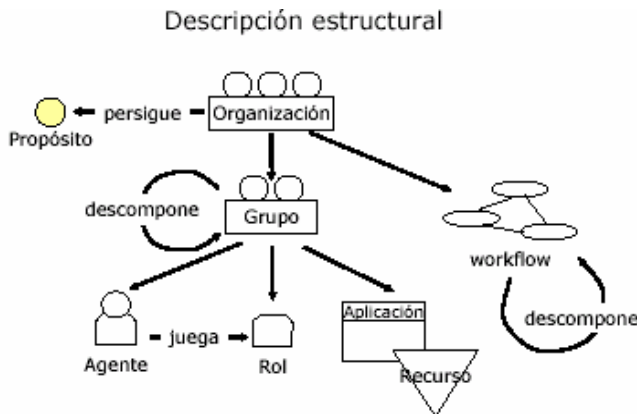
La organización, desde el punto de vista estructural, es un conjunto de entidades asociadas por relaciones simples de agregación y herencia. En esta estructura se define el esqueleto donde van a existir los agentes, los recursos, tareas y objetivos. Sobre este esqueleto se definen una serie de relaciones que inducen la formación de flujos de trabajo y de restricciones sociales

Además la organización es una entidad autónoma, como lo es también un agente. Los objetivos perseguidos por la organización son los objetivos comunes a los agentes que la componen y el motivo por el cual se han agrupado. No obstante, una organización no es un agente. La diferencia fundamental es que la organización no tiene capacidad de ejecutar tareas ni para tomar decisiones, son los agentes que la componen quienes se encargan de ello.

---

<sup>38</sup> Web Oficial Metodología: <http://grasia.fdi.ucm.es/ingenias/Spain/lenguaje/morganizacion.htm>

La organización se descompone en: Grupos y Flujos de Trabajo. Cada *grupo* contiene agentes, recursos, aplicaciones o roles. La asignación de estos elementos a un grupo obedece a propósitos organizativos, esto es, están agrupados porque clarifica la creación de flujos de trabajo. Cualquiera de ellos puede pertenecer a otros grupos de la misma u otras organizaciones.



**Figura 2.10 Descripción estructural**

Para aumentar la capacidad de abstracción e incrementar la reusabilidad de los agentes y roles, se permite una herencia simple entre agentes, mientras que con los roles se admite herencia múltiple. La semántica de la herencia es parecida a la semántica de la herencia de objetos.

## 2.6.2.- DESCRIPCIÓN SOCIAL

Las relaciones sociales entre Organizaciones, Agentes y Grupos configuran restricciones sociales que limitan la interacción entre estas entidades. Aquí se incluyen relaciones de:

- Subordinación
- Prestación de un servicio
- Cliente de un servicio.

Las relaciones de subordinación obligan al subordinado a obedecer en todo y permiten al subordinante dar cualquier orden. El siguiente paso es descomponer esta relación en obediencia incondicional y condicionada. La incondicional consiste en obedecer todas y cada una de las órdenes. La condicional es asimilable a un contrato donde, si se infringen algunas de las condiciones impuestas, el contrato deja de tener validez. Las condiciones del contrato se establecen utilizando un Patrón de estado mental.

Respecto a las relaciones Proveedor y Cliente, hacen referencia a que existe un servicio ofrecido (proveedor) y que se busca un servicio determinado (cliente). A la hora de pedir o informar de la capacidad de dar o recibir un servicio, simplemente se informa qué tareas se puede realizar o qué se busca utilizando un objetivo. La técnica es similar a la más extendida de definición de ontología. La ontología la constituyen los objetivos a satisfacer.

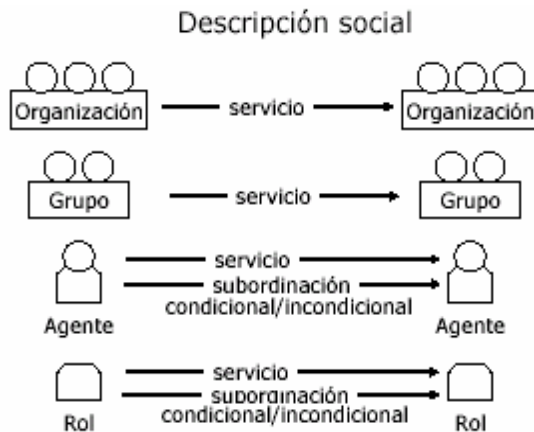


Figura 2.11 Descripción social

En cierto modo la subordinación condicional es similar a la prestación de un servicio, ya que en ambos se busca ejecutar una acción para otro. Sin embargo, son diferentes en la forma. En la prestación de un servicio, el proveedor puede denegar la petición, mientras que en la subordinación existe la obligación de ejecutar lo pedido.

### 2.6.2.1.- Prioridad entre las relaciones

Entre las distintas entidades de la organización surgen dieciséis posibles combinaciones de dependencias entre organizaciones, grupos, roles y agentes. Para limitar y estructurar estas relaciones, se admiten sólo relaciones entre entidades del mismo tipo. Se establece que entre dos entidades de la organización pueden existir como máximo relaciones a tres niveles. De mayor a menor nivel, estos niveles son:

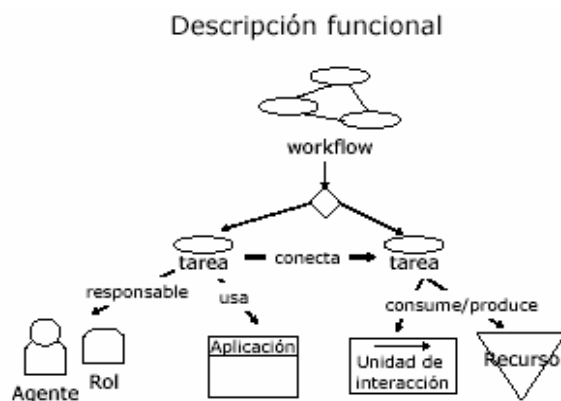
- Nivel organizativo: Se trata de las relaciones organizacionales. Se dan entre las entidades de más alto nivel y son el equivalente a las relaciones interempresariales.
- Nivel de estructura de organización: Se trata de las relaciones estructurales. Serían las equivalentes a relaciones interdepartamentales.
- Nivel de agente y rol: Se trata de las relaciones de agencia. Serían equivalentes a relaciones directas entre el personal.

No puede haber relaciones sociales entre dos entidades entre las cuales no exista una relación a nivel inmediatamente superior.

### 2.6.3.- DESCRIPCIÓN FUNCIONAL

La definición funcional establece qué ofrece la organización y cómo se lleva ésta a cabo. Aquí, el objetivo del flujo de trabajo es establecer cómo se asignan los recursos, qué pasos (tareas) son necesarios para la consecución de un objetivo, y quiénes son los responsables de ejecutarlas. En el flujo de trabajo, se habla de actividades en lugar de tareas, aunque en este contexto se pueden emplear indistintamente.

Dentro del *flujo del trabajo* interesa presentar dos tipos de información: cómo se asocian unas tareas con otras y cómo se ejecutan. Se permite que un flujo de trabajo se descomponga en otros flujos de trabajo o tareas. De esta forma se pueden generar definiciones incrementales de las tareas que componen el flujo de trabajo. No se permite la descomposición de una tarea en flujos de trabajo.



Las tareas son ejecutadas por agentes que son directamente responsables de ellas o indirectamente a través de roles. Que un agente o rol sea responsable de la ejecución de una tarea, significa que el agente sabe ejecutar esa tarea.

Figura 2.12 Descripción funcional

### 2.7.- META – MODELO DE ENTORNO

Esa metodología modela el entorno utilizando un conjunto finito de variables observables.<sup>39</sup> Discretiza el entorno en dos direcciones: categorizar el tipo de

<sup>39</sup> Web Oficial Metodología <http://grasia.fdi.ucm.es/ingenias/Spain/lenguaje/mentorno.htm>

entidades relevantes en el entorno y restringir la interacción con las mismas. Así, el entorno contendrá sólo recursos, aplicaciones y agentes, y se limitará la percepción y actuación de los agentes.

Las aplicaciones en INGENIAS pueden modelarse de dos formas: abstracciones con objetos y con agentes.

Usando objetos: Las aplicaciones se recubren con una capa de objetos. La interacción con estos componentes se define en términos de conceptos de UML (interfaces, diagramas de estados, diagramas de colaboración). Con esta solución, se integran subsistemas como sistemas propietarios o bases de datos, definiendo una interfaz y un comportamiento asociado al objeto.

Usando agentes: Por medio de tres formas de convertir en agente: por transductor (*transducer*), mediante recubrimiento (*wrapping*) y reescritura. El primero consiste en modelar la entidad como un objeto y asociar este objeto con un agente, el cual se encarga de mediar con el resto de agentes. El segundo consiste en incrustar la entidad dentro de un agente, de tal forma que el agente redirija las peticiones a la propia entidad. El tercero consiste básicamente en reescribir la aplicación.

Dependiendo del dominio, una tendrá ventajas sobre las otras. La experiencia dicta que la solución menos costosa es la primera.

Por último, se considera la posibilidad de que existan otros agentes en el sistema. Para este caso, el comportamiento de estos agentes es modelable con los meta-modelos presentados en esta metodología.

Las aplicaciones además de servir como actuadores y sensores de los agentes, se utilizan para integrar software en el desarrollo del SMA. Pueden emplearse para modelar servicios pasivos, esto es, un conjunto de operaciones que no requiere la interacción con ningún agente y que son empleadas por varios agentes. Estos servicios pueden existir previamente en el entorno o no. De forma más frecuente, servirían como interfaz con el

mundo real. Así la generación de estas aplicaciones estaría relacionada con el estudio del entorno para obtener un conjunto de variables de entrada, que representan datos accesibles en un momento dado mediante dos tipos de percepción: muestreo (*polling*) o por disparo de interrupciones (en ambos tipos se asocia la percepción (*EPercibe*) con una operación concreta), y salida, que sirven generalmente como parámetros de las acciones a ejecutar como reacción del sistema ante los cambios del entorno. Estas variables, aquí aparecerían como métodos asociados a las aplicaciones, para lectura de datos o actuación sobre el entorno.

Las aplicaciones pueden existir con anterioridad al desarrollo actual (*AplicacionEntorno*) o ser desarrolladas para los propósitos actuales (*AplicacionInterna*). Las primeras se obtienen de la captura de requisitos, mientras que las segundas se generan mediante técnicas convencionales de ingeniería del software.

Por recurso se entiende todo aquel objeto del entorno que no proporciona una funcionalidad concreta, pero que es indispensable para la ejecución de tareas y cuyo uso se restringe a consumir o restituir. Los recursos se categorizan en:

- Consumibles. Son recursos que con el uso se agotan pero que son restituibles.
- No consumibles. El recurso se auto-restituye al término de la acción realizada sobre él.

Todos los recursos pueden pertenecer a un agente o ninguno (el que lo usa es el poseedor temporal). La capacidad de estos recursos se define con tres atributos del mismo tipo (real, entero, tipo compuesto). Los atributos son *estado*, *nivel de uso mínimo* y *nivel de sobrecarga*. El uso de un recurso hace que su estado decrezca hasta el nivel de uso mínimo, a partir del cual el recurso se inhabilita. Otra situación de inhabilitación es el restablecimiento excesivo del recurso, que incrementa su estado por encima del nivel de sobrecarga.

En la siguiente tabla se presenta las posibles instancias de recursos.

<b>Descripción del recurso</b>	<b>Atributos</b>	<b>Tipo</b>
Descriptores de ficheros	Número máximo, número utilizado	No-consumible
Hilos de ejecución	Número máximo, número utilizado	No-consumible
Memoria	Máxima disponible, memoria utilizada	No-consumible
Dispositivos de salida	Número de dispositivos disponibles, cantidad utilizada, exclusividad	Consumible
Dispositivos de entrada	Número de dispositivos disponibles, cantidad utilizada, exclusividad	Consumible
Unidad de almacenamiento	Espacio libre, espacio utilizado	Consumible
Sockets	Máximos disponibles, número utilizados	No-consumible
Ancho de banda	Máximo disponibles, número utilizados	No-consumible

**Tabla 2.3 Enumeración de ejemplos de recursos**

Las tareas se relacionan con los recursos mediante tres relaciones: consumición, producción y limitación. La relación de producción permite que una tarea restituya un recurso. La de limitación impone precondiciones para lanzar una tarea en función de la disponibilidad de recursos. Finalmente, la de consumición indica uso de un recurso, lo que conlleva una disminución del estado del recurso.

Por último, la denominación de agente se emplea cuando la entidad satisfaga el principio de racionalidad.

## **2.8.- CICLO DE VIDA**

Dentro de esta metodología, desde el punto de vista de ingeniería, los meta-modelos sirven de guía, y muestran cómo pueden ayudar a estructurar el desarrollo de SMA. Para estudiar la aplicación de los meta-modelos a procesos de ingeniería, se han tomado los meta-modelos como lenguaje de especificación del SMA, de forma similar a los diagramas UML como especificación de un desarrollo orientado a objetos.<sup>40</sup>

<sup>40</sup> Web Oficial Metodología <http://grasia.fdi.ucm.es/ingenias/Spain/integracion/index.php>

INGENIAS integra los meta-modelos en el Rational Unified Process (*RUP*) en las fases de análisis y diseño, por lo tanto su ciclo de vida es iterativo e incremental, utiliza casos de uso para determinar la funcionalidad del sistema e interacciones como generalización de diagramas de colaboración y diagramas de secuencia. Para orientar la integración, se plantean una serie de asociaciones entre elementos del RUP y elementos de los meta-modelos.

<b>Entidad MAS GRASIA</b>	<b>Entidad RUP</b>
Agente	Clase
Organización	Arquitectura
Grupo	Subsistema
Interacción	Escenario
Roles, tareas y flujos de trabajo	Funcionalidad

**Tabla 2.4 Asociaciones entre los elementos del RUP y entidades de MAS GRASIA**

El agente, como la clase, define tipos. Lo que aquí se ha denominado agente en ejecución sería un objeto en el RUP. La organización equivale a la arquitectura en el RUP por su carácter estructurador. La organización da una visión global del sistema agrupando agentes, roles, recursos y aplicaciones y estableciendo su participación en los flujos de trabajo del SMA. El grupo es la unidad de estructuración utilizada en la organización. Su similitud con un subsistema se debe a que como éste, se utiliza para organizar elementos en unidades de abstracción mayores y define un conjunto de interfaces para interaccionar los roles, en este caso. La interacción, se ve como una generalización de los diagramas de colaboración y secuencia. En el RUP, los diagramas de secuencia y colaboración se ven como escenarios que describen cada caso de uso. Por último, roles, tareas y los flujos de trabajo proporcionan el encapsulamiento de acciones que en el RUP dan métodos e interfaces.

En el RUP, el esfuerzo del análisis y diseño se encuentra localizado en tres fases: inicio, elaboración y construcción. Dentro de cada fase se desarrollan las iteraciones (ciclos completos de desarrollo incluyendo análisis, diseño, implementación y pruebas) que construyen gradualmente el sistema. En la tabla 2.5 se muestra la adecuación de las etapas



del RUP a los meta-modelos presentados en esta metodología de acuerdo con estas equivalencias y el proceso del RUP.

		<b>FASES</b>		
<b>F L U J O S</b>	<b>D E S I G N O</b>	<b>Inicio</b>	<b>Elaboración</b>	<b>Construcción</b>
		<b>A</b>	<ul style="list-style-type: none"> <li>- Generar casos de uso e identificar realizaciones de los casos de uso con modelos de interacciones.</li> <li>- Esbozar la arquitectura con un modelo de organización.</li> <li>- Generar modelos del entorno para trasladar la captura de requisitos a los modelos</li> </ul>	<ul style="list-style-type: none"> <li>- Refinar casos de uso.</li> <li>- Generar modelos de agente para detallar los elementos de la arquitectura.</li> <li>- Continuar con los modelos de organización identificando flujos de trabajo y tareas.</li> <li>- Modelos de tareas y objetivos para generar restricciones de control (objetivos principales, descomposición de objetivos)</li> <li>- Refinar modelo de entorno para incluir nuevos elementos.</li> </ul>
<b>J O S</b>	<b>D</b>	<ul style="list-style-type: none"> <li>- Generar un prototipo con herramientas de prototipado rápido, como ZEUS o Agent Tool</li> </ul>	<ul style="list-style-type: none"> <li>- Centrar el modelo de organización en el desarrollo de flujos de trabajo.</li> <li>- Llevar las restricciones identificadas a modelos de tareas y objetivos para dar detalles acerca de las necesidades y resultados de las tareas y su relación con los objetivos del sistema.</li> <li>- Expresar la ejecución de tareas dentro de modelos de interacción.</li> <li>- Generar modelos de agente para detallar patrones de estado mental.</li> </ul>	<ul style="list-style-type: none"> <li>- Generar nuevos modelos de agente o refinar los existentes.</li> <li>- Depurar la organización centrandolo el desarrollo en las relaciones sociales.</li> </ul>

**Tabla 2.5 Adecuación de las etapas del RUP a los meta-modelos presentados.**

Las fases de pruebas e implementación no se han incluido. La fase de pruebas es igual a la del software convencional. En cuanto a la implementación, es posible la generación automática de código basada en XML a partir de la especificación de los meta-modelos. Con ello se consigue que sea independiente de la plataforma de agentes o del lenguaje de agentes utilizado. También es posible, en esta etapa, que el diseñador pueda actuar como en diseños usuales, desarrollando elementos computacionales que hagan lo que está especificado.

A continuación se presenta en forma general los resultados que se obtienen durante el análisis y el diseño en cada meta-modelo. La generación de modelos a partir de éstos meta-

modelos se guía por un conjunto de actividades para cada uno de ellos, las cuales se puede revisar detalladamente en el sitio web.

### **2.8.1.- GENERACIÓN DE INSTANCIAS DEL META-MODELO DE AGENTE**

Los resultados que se obtienen durante el análisis son los siguientes:

- Funcionalidad del agente: Se identifica los roles que desempeña cada agente y las tareas que deben realizar según como se asocia, al agente o al rol.
- Requisitos del agente: Se especifica qué tipo de inteligencia y qué tipo de autonomía se desea que el agente demuestre.

Los resultados a obtener durante el diseño se refieren al control del agente:

- Restricciones de control: Se debe representar los estados mentales por los que pasa el agente a lo largo de las interacciones.

### **2.8.2.- GENERACIÓN DE INSTANCIAS DEL META-MODELO DE INTERACCIONES**

En el análisis los resultados esperables son:

- Interacciones relevantes en el sistema: Se debe indicar participantes y objetivos perseguidos.
- Esquema inicial de intercambio de mensajes: Se realiza con diagramas de colaboración o de secuencia UML

Para el diseño, los resultados esperables son:

- Descripciones detalladas de las interacciones: El nivel de detalle es en función de las entidades que se escogen para implementar las interacciones.
- Contexto de las interacciones: Las interacciones contextualizadas en los flujos de trabajo de la organización y con información adicional acerca de las condiciones de participación en la interacción.

- Conjunto de unidades de interacción asociadas a los participantes: Cada participante en las interacciones comparte con las otras unidades de interacción. Esto forma parte de la descripción funcional del agente.

### 2.8.3.- GENERACIÓN DE INSTANCIAS DEL META-MODELO DE TAREAS Y OBJETIVOS

Para el análisis los resultados esperables son:

- Tareas y objetivos: Los objetivos que se persiguen en el sistema, que han de ser resumen de todos los incluidos en los modelos de agente y organización. Se admite la descomposición estructural de tareas y objetivos (*GTDescompone* y *WFDescompone*) para disminuir su complejidad. Esta descomposición se acompaña de dependencias entre objetivos (*GTDepende*)
- Tareas asociadas a objetivos: La asociación consiste inicialmente en instancias de *GTAfecta* que evolucionarán en el diseño hacia *GTSatisface* o *GTFallas*. Esta asociación constituye la justificación de la ejecución de la tarea.
- Precondiciones y postcondiciones tentativas: Las precondiciones mínimas son determinar qué entidades mentales se consumen (*WFConsume*), qué interacciones y entidades mentales se producen (*WFProduce* y *GTCrea*) y qué aplicaciones se usarán en el proceso (*WFUsa*).

Del diseño se espera:

- Refinamiento de las dependencias entre objetivos: Se detalla la dependencia indicando si se trata de dependencia *Y* (*GTDependeY*) o dependencia *O* (*GTDependeO*). Las dependencias pueden desaparecer con motivo de cambios en las dependencias estructurales entre objetivos.
- Condiciones de satisfacción o fallo de los objetivos: Las relaciones *GTAfecta* entre tareas y objetivos evolucionan a *GTFallas* y *GTSatisface*. En estos dos casos es necesario además indicar bajo qué condiciones se asume el éxito o fracaso del objetivo. Estas condiciones se expresan con *patrones de estado mental*.
- Precondiciones detalladas: El detalle en las precondiciones viene de añadir los recursos que necesita la tarea (*WFUsa*) y de incluir nuevas instancias de

*WFConsume*, *GTDestruye*, *GTModifica*. Estas dos últimas se interpretan como precondiciones en el sentido de que requieren la existencia de la entidad mental con la que se relaciona.

- Postcondiciones detalladas: Se asocian nuevas instancias *WFProduce* con recursos para expresar la restitución de los mismos. Adicionalmente, se incluyen instancias *WFUsaLlamada* que expresan qué operaciones se están utilizando de las aplicaciones asociadas. Asimismo, se incluyen instancias de *GTDestruye*, *GTModifica* y *GTCrea* para mostrar cómo se modifica el estado mental. Se pueden asociar también entidades mentales con *WFProduce*, pero en dicho caso se interpreta la producción en términos de flujo de trabajo (información pasada de una tarea a otra).

#### **2.8.4.- GENERACIÓN DE INSTANCIAS DEL META-MODELO DE ORGANIZACIÓN**

Los resultados que se obtienen durante el análisis son los siguientes:

- Estructura de la organización: Se identifica los elementos del sistema, se debe detallarlo lo mayor posible.
- Flujos de trabajo: Se debe generar una lista de las tareas, con sus responsables correspondientes, conectadas unas con otras y resaltando aquellas tareas cuya ejecución afecte a otros agentes.
- Dependencias sociales: En el análisis basta con tejer una red de instancias de *AGORelacion*.

Para el diseño se ha fijado que los resultados esperables son:

- Contexto de ejecución de las tareas: Las tareas se asocian con sus ejecutores para indicar quien se ve afectado por ellas, qué recursos y entidades mentales se consumen y qué entidades de información se producen.
- Refinamiento de las dependencias sociales: Hay que detallar qué tipo de relaciones se están definiendo cuando se interconectan dos entidades con *AGORelacion*. Concretamente, se debe determinar si hay o no subordinación o si

se trata de una relación cliente-servidor, indicando en cada caso las condiciones de prestación de servicio y el servicio en sí.

#### **2.8.5.- GENERACIÓN DE INSTANCIAS DEL META-MODELO DE ENTORNO**

Para el análisis los resultados esperables son:

- Entidades del entorno: Una enumeración de las entidades que preceden al sistema a desarrollar. Estas entidades se categorizan como aplicaciones, agentes o recursos. Cada entidad se acompaña de una descripción de sus funciones, que se entiende han de extraerse de la captura de requisitos.
- Percepción del agente: La percepción del agente se asocia con las aplicaciones del entorno.

Para el diseño se ha fijado que los resultados esperables son:

- Configuración de la percepción del agente: Cada agente asociado con aplicaciones para determinar cómo va a percibir el entorno, si va a utilizar mecanismos de muestreo o notificación y qué es lo que espera recibir.
- Relaciones con tareas: Aunque se considera en otros meta-modelos, se identifican asociaciones de producción y utilización de aplicaciones y recursos con tareas.
- Definición detallada de los recursos: Consiste en especificar exactamente cómo es cada recurso, detallando su categoría concreta, el límite inferior de consumo, el superior y el estado en que se encuentra inicialmente.

## CAPÍTULO III

### III.- PLATAFORMA JADE

#### 3.1.- INTRODUCCIÓN

JADE (*Java Agent DEvelopment Framework*) es un middleware desarrollado por TILAB (Telecom Italia)<sup>41</sup> para el desarrollo de aplicaciones distribuidas multiagente, que proporciona, tanto un entorno de desarrollo como un entorno de ejecución, para la realización y mantenimiento de sistemas multiagente.

JADE responde a los estándares FIPA (organización de sistemas multiagente y de comunicación entre agentes visto en el Capítulo I), esto quiere decir que las especificaciones que incluye la arquitectura abstracta se van a encontrar también aquí.

Esta plataforma provee un entorno de desarrollo formado por una serie de librerías en Java que permiten la implementación de agentes de manera limpia e independiente de la plataforma sobre la que se va a ejecutar. Además provee un entorno de ejecución gráfico que permite a los agentes vivir y comunicarse entre ellos. Está realizado enteramente en Java y proporciona una serie de herramientas que permiten al desarrollador controlar y depurar a los agentes en tiempo real.

De entre otros middlewares similares,<sup>41</sup> JADE está siendo el más utilizado y se presenta como el estándar de facto en su tipo. Algunas razones de esto son: se apega a las especificaciones de FIPA, es de libre distribución, su código fuente es abierto.

---

<sup>41</sup> Es una capa de software que busca implementar servicios genéricos para sistemas distribuidos. Estos servicios principales son: identificación, autenticación, autorización, directorios, seguridad, servicios generales de red, API normalizado, promover interoperabilidad, facilitar construcción de sistemas.

### 3.2.- ARQUITECTURA

Los Agentes JADE necesitan del entorno de ejecución donde poder "vivir", cada instancia del entorno de ejecución se denomina contenedor (*container*), el conjunto de contenedores conforman una plataforma (*platform*) y así se provee una capa homogénea para los agentes, esto es el desarrollo de las aplicaciones se hará sin importar hardware, sistema operativo, tipo de red, o JVM (*Java Virtual Machine*).<sup>42</sup>

En cada plataforma debe existir un contenedor especial denominado contenedor principal (*main container*). La principal diferencia del contenedor principal respecto al resto de contenedores es que alberga dos agentes especiales: AMS (*Agent Management System*), DF (*Directory Facilitator*) (definidos en el estándar FIPA visto en el capítulo 1). La arquitectura se puede observar en la figura 3.1, donde aparecen dos plataformas diferentes (*platform1*, *platform2*).

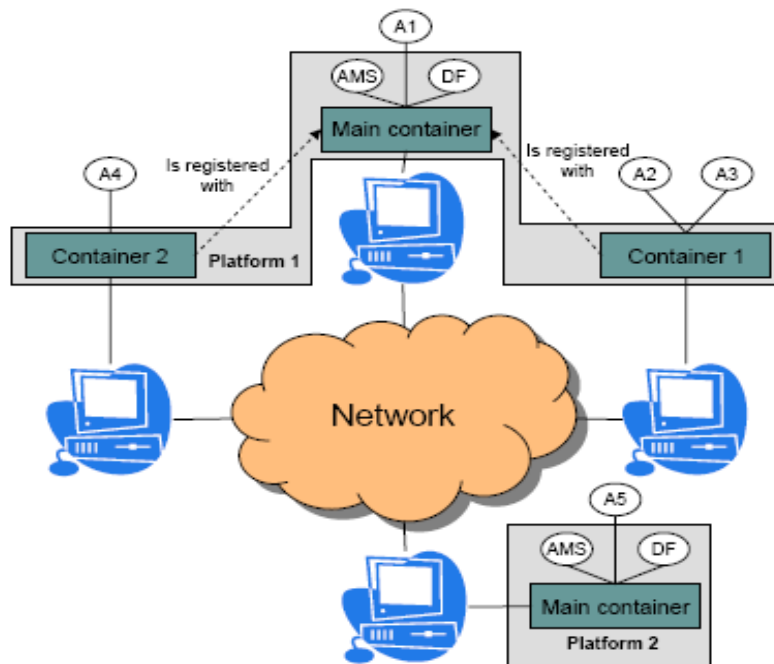


Figura 3.1 Esquema de distribución de los contenedores y las plataformas

<sup>42</sup> Introducción a JADE: [www.upv.es/sma/practicas/jade/Introducci%F3nJADE.pdf](http://www.upv.es/sma/practicas/jade/Introducci%F3nJADE.pdf)

En la primera plataforma se tiene a los agentes A2 y A3 pertenecientes al container 1, en el container 2 al agente A4 y en el main container los agentes AMS, DF y A1. En cambio en la segunda plataforma se tiene solamente un contenedor principal con los agentes AMS, DF y A5. En base a esta distribución, se permite la comunicación entre agentes de un container, entre agentes de varios containers y entre agentes de diferentes plataformas mediante comunicaciones punto a punto.

Los agentes JADE tienen nombres únicos. Los agentes proporcionan servicios, cada agente puede buscar a otros dependiendo de los servicios que proporcionen otros agentes.

La comunicación entre agentes se lleva a cabo a través de mensajes asíncronos, es decir, el agente que envía el mensaje y el destinatario del mensaje no tienen porqué estar disponibles al mismo tiempo. Es más, el destinatario no tiene porqué existir en ese instante. Los mensajes se pueden enviar a un agente en concreto o se pueden enviar a agentes que se desconocen pero se sabe que poseen ciertas características. JADE proporciona mecanismos de seguridad, ya que habrá agentes a los que no se les esté permitido comunicarse con otros agentes, de manera que una aplicación puede verificar la identidad del receptor y del agente que envía el mensaje y no dejar realizar actuaciones no permitidas para un determinado agente. La estructura de los mensajes se basa en el lenguaje ACL (*Agent Communication Language*) que ha sido definido por la FIPA (*Foundation for Intelligent Physical Agents*).

JADE también permite a los agentes cambiar de Host (en J2SE Java 2 Stándar Edition). Un agente puede interrumpir en un momento dado su ejecución, migrar a otro host (sin necesidad de que el código esté previamente en el host destino) y continuar la ejecución en el mismo punto en el que la interrumpió. Esto permite un balanceo de carga ya que permite a los agentes migrar hosts menos cargados. El entorno de ejecución proporciona un marco donde poder ejecutar los agentes y herramientas gráficas para su monitorización y depuración.

JADE también presenta las siguientes características:



- P2P: Arquitectura peer-to-peer, cada agente puede tomar la iniciativa en una comunicación o bien responder a peticiones que le hagan otros agentes.
- Interoperabilidad: JADE cumple con las especificaciones FIPA, por lo que los agentes desarrollados en JADE pueden interactuar con otros agentes que no tienen porque estar desarrollados con JADE, aunque si deben seguir las especificaciones FIPA.
- Intuitiva: JADE se ha desarrollado para ofrecer una API (*Application Programming Interface*) fácil de aprender y sencilla de manejar.
- Concurrente: Los agentes funcionan mediante hebras.
- Ciclo de vida: Controla el ciclo de vida de los agentes (crear, suspender, reanudar, matar)
- Portabilidad: La API (*Application Programming Interface*) que proporciona JADE es independientemente de la red sobre la que va a operar, así como de la versión de Java utilizada, teniendo la misma API para J2EE, J2SE y J2ME, como se puede ver en la figura 3.2

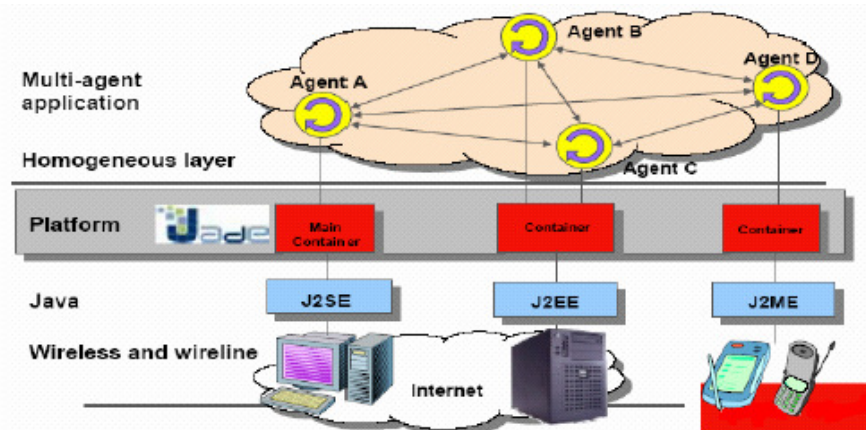


Figura 3.2 Portabilidad de JADE

JADE particularmente puede ser aplicado en ambientes de dispositivos móviles, teléfonos o PDAs integrando las redes inalámbricas junto con las redes convencionales. Para ello surge LEAP (*Lightweight Extensible Agent Platform*) que permite ejecutar agentes JADE en dispositivos móviles y/o conectados a través de redes inalámbricas

### 3.3.- ENTORNO DE EJECUCIÓN

Para ejecutar un programa agente en JADE basta con hacerlo desde la línea de comandos. JADE también proporciona un conjunto de agentes que realmente actúan a modo de plataforma, de manera que a partir de ellos se puede ver como los programas interactúan. Los agentes de los que se habla son el agente *RMA*, el agente *Dummy*, el agente *Sniffer*, el *DF GUI*, agente *Instrospector* que se detallaran más adelante.<sup>43</sup>

#### 3.3.1.- INSTALACIÓN

Al ser JADE una plataforma desarrollada en Java será posible utilizarla en cualquier sistema operativo tanto en sistemas Windows como en sistemas Linux o BSD. Se debe tener instalado previamente Java Run Time Environment 1.4 (*JRE*) y el kit de desarrollo de Java (*JDK*).

La versión 3.2 de JADE (*JADE-all-3.2.zip*) puede descargarse de la siguiente dirección <http://jade.tilab.com>. Este archivo a su vez, alberga los siguientes cuatro archivos que aparecen a continuación:

- *JADE-doc-3.2.zip*: la documentación javadoc, el manual del administrador, el del programador y un tutorial.
- *JADE-src-3.2.zip*: el código fuente sin compilar.
- *JADE-bin-3.2.zip*: el código ya compilado y listo para ser interpretado.
- *JADE-examples-3.2.zip*: ejemplos de uso de la plataforma.

Una vez obtenido el archivo *JADE-all-3.2.zip*, se crea un directorio a parte para JADE, se coloca este fichero y se lo desempaqueta. En seguida se creará un directorio *jade* y bajo este directorio un directorio *lib* que es en donde están los *jars* necesarios para ejecutar la plataforma.

Se debe definir variables de entorno:

---

<sup>43</sup> Tutorial Básico de JADE: [galahad.plg.inf.uc3m.es/~docweb/swagent/jade\\_texto.pdf](http://galahad.plg.inf.uc3m.es/~docweb/swagent/jade_texto.pdf)

### En Windows

JADE\_HOME = c:\jade

CLASSPATH =

.;%JADE\_HOME%\lib\jade.jar;%JADE\_HOME%\lib\jadeTools.jar;%JADE\_HOME%\lib\iiop.jar;%JADE\_HOME%\lib\commons-codec\commons-codec-1.3.jar;.....

### En Linux

JADE\_HOME = /usr/local/jade

CLASSPATH =

.\$JADE\_HOME\lib\jade.jar;\$JADE\_HOME\lib\jadeTools.jar;\$JADE\_HOME\lib\iiop.jar:\$JADE\_HOME\lib\commons-codec\commons-codec-1.3.jar .....

## 3.3.2.- EJECUCIÓN

Para iniciar la plataforma JADE, se dispone de un comando:

```
java jade.Boot [options] [AgentSpecifier list]
```

Opciones principales:

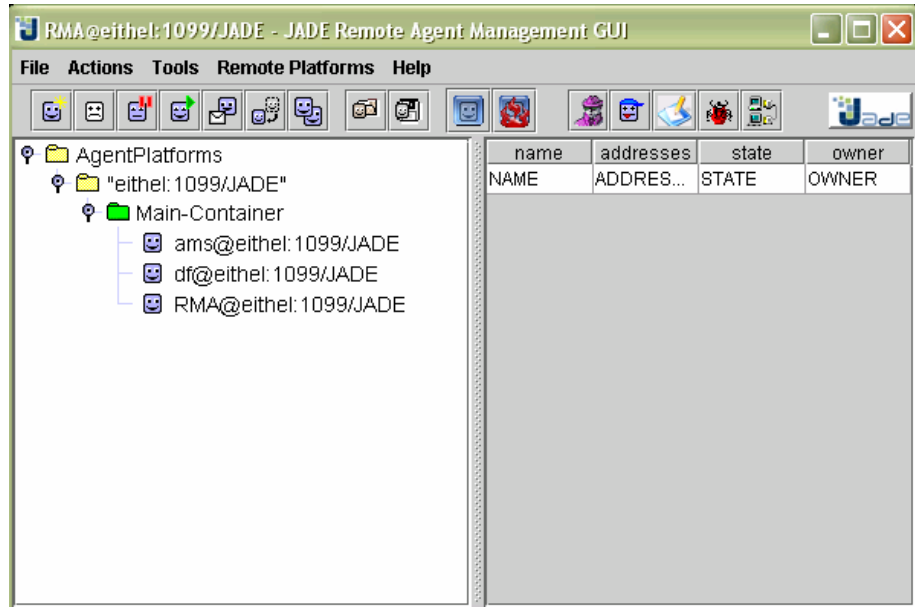
- container (si es distinto del principal)
- host (nombre del host)
- port (puerto, por defecto es el 1099)
- name (nombre simbólico de la plataforma)
- gui (lanza el RMA)
- mtp (permite añadir protocolos externos además del IIOP (Internet Inter-ORB Protocol))

En este caso se desea ver la interfaz gráfica RMA, dentro del directorio donde se haya descomprimido JADE. Se debe teclear lo siguiente: **java jade.Boot -gui**

RMA (Remote Monitoring Agent): Permite controlar el ciclo de vida de todos los agentes residentes en la plataforma. Su función es crear, suspender y matar un agente, así como clonarlo y moverlo hacia otro contenedor. Se lanza poniendo la opción -gui cuando

se ejecuta un ejemplo escrito en Java o también puede ser lanzado desde la línea de comandos java:

```
jade.Boot myConsole;jade.tools.rma.rma.
```



**Figura 3.3 Interfaz gráfica del RMA de JADE**

En la figura 3.3 se puede apreciar la interfaz del RMA. En la parte inferior se muestra, a la izquierda, el árbol formado por plataformas, contenedores y agentes, y a la derecha, las características del agente seleccionado. Estos conceptos han sido estudiados anteriormente en la arquitectura de JADE. Existe un contenedor principal (que alberga dos agentes el AMS y DF) y cuya función es almacenar agentes, no necesariamente en el mismo ordenador. Los agentes utilizan el protocolo RMI para comunicarse entre varios contenedores.

Los agentes deben tener nombres únicos. Un nombre se compone de un alias o nickname y una dirección separados por el signo @. Por ejemplo RMA@PC2:1099/JADE es un agente con nickname RMA en la dirección PC2:1099/JADE (PC2 es el nombre de la máquina en la LAN y 1099 el puerto utilizado, estudiado en el estándar FIPA Capítulo I)

En la parte superior de la interfaz se encuentra la barra de menú:

*File:*

- *Close RMA Agent*: cierra la interfaz anteriormente mostrada, aunque la plataforma de agentes continuará funcionando, así como todos aquellos agentes que estuvieran activos.
- *Exit this container*: Destruye el container donde el agente RMA se aloja, matando a este agente y todos aquellos que le acompañaran dentro del container. Si dicho container es el MainContainer será toda la plataforma la que se desactivará.
- *Shutdown Agent Platform*: cierra totalmente la plataforma, eliminando todos los agentes y containers.

*Actions:*

- *Start New Agent*: esta acción crea un agente. Al usuario se le preguntará por el nombre del agente y la clase a la cual se instanciará (los agentes son instancias de alguna clase). Si hubiera algún container seleccionado, éste será en el que dicho agente se alojará. También se puede indicar el nombre del container en el que se desea que se ejecute el agente.
- *Kill*: mata a todos los agentes y elimina todos los containers seleccionados. Si el MainContainer estuviera seleccionado, la plataforma entera sería cerrada.
- *Suspend*: suspende los agentes seleccionados.
- *Resume*: este comando vuelve a poner en estado de activo los agentes seleccionados que estuvieran suspendidos.
- *Change owner*: permite cambiar el dueño de un agente, sin tener que reiniciar la plataforma. Se solicita el nombre del usuario así como su contraseña.
- *Send Message*: este comando permite enviar un mensaje ACL (Agent Communication Language, visto en el capítulo I) a un agente.
- *Migrate Agent*: esta acción permite migrar un agente. Cuando se selecciona, un diálogo es mostrado, en el que el usuario debe especificar el container de la plataforma donde el agente seleccionado debe migrar. No todos los agentes pueden ser migrados debido a una falta de serialización en su implementación.

- *Clone Agent*: esta opción permite clonar un agente. Cuando se selecciona un diálogo se muestra dónde el usuario debe introducir el nombre del nuevo agente y el container donde será alojado.

*Tools:*

- *Start Sniffer*: crea un agente de tipo Sniffer.
- *Start DummyAgent*: crea un agente de tipo DummyAgent.
- *Show the DF GUI*: muestra la interfaz asociada al agente DF, del que ya se ha hablado anteriormente.
- *Start introspectorAgent*: inicia un agente de tipo introspectorAgent, mostrando su interfaz asociada.

*Remote Platforms:*

- Este menú permite controlar plataformas remotas que cumplan con las especificaciones FIPA. Estas plataformas no tienen por qué ser JADE.

*Help:*

- *About JADE*: cuadro de *Acerca de* que muestra las características de la plataforma.

Debajo de la barra de menús se dispone de una serie de botones que equivalen a algunos de los comandos de dichos menús.

*Agente Dummy*: Permite componer y mandar mensajes *ACL* así como mantener una lista con todos los mensajes *ACL* mandados y recibidos. Esta lista puede ser examinada por el usuario, por lo que cada mensaje se puede ver con detalle. Dicha lista puede ser también grabada y recuperada más tarde. Al nuevo DummyAgent se le asignará automáticamente un nombre.

Puede ser lanzado como un agente normal al escribir en la línea de comandos:

```
java jade.Boot theDummy:jade.tools.DummyAgent.DummyAgent.
```

Al crear el agente se mostrará una ventana en la que se puede acceder a todas las funcionalidades del mismo:

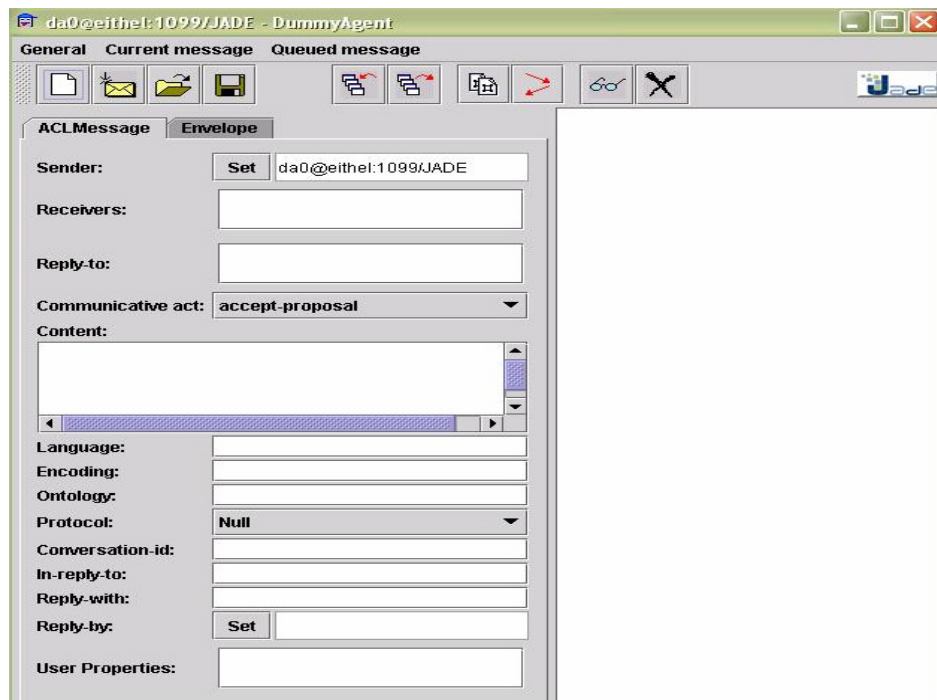


Figura 3.4 Interfaz gráfica del Agente Dummy

La parte principal de la ventana está dividida en dos. En la izquierda se dispone de una serie de campos donde se puede introducir el valor de los campos del mensaje a enviar. En la parte derecha se muestra la cola de mensajes enviados y recibidos. Con respecto a los menús, a continuación se muestra la función de cada una de sus opciones:

*General:*

- *Exit:* cierra el diálogo del agente DummyAgent, eliminándolo de la plataforma.

*Current message:*

- *Reset:* vacía todos los campos del mensaje a enviar.
- *Send:* envía el mensaje con los valores de los campos especificados en la parte izquierda del diálogo. Al hacerlo, se mostrará el mensaje enviado en la cola de mensajes en la parte derecha de la ventana, así como las contestaciones, en el caso de que las hubiera.
- *Open:* abre un mensaje almacenado en disco.
- *Save:* guarda el mensaje en disco.

*Queued message:*

- *Open queue:* abre una cola de mensajes almacenada en disco.
- *Save queue:* guarda la cola de mensajes en disco.
- *Set as current:* toma los valores de los campos del mensaje seleccionado en la cola de mensajes y rellena los campos de la parte izquierda de la ventana correspondientes al mensaje a enviar con dichos valores.
- *Reply:* permite crear un mensaje de respuesta.
- *View:* muestra un diálogo donde se podrán examinar los valores de los campos del mensaje seleccionado de la cola de mensajes.
- *Delete:* elimina el mensaje seleccionado de la cola de mensajes.

*Agente Sniffer:* Permite seleccionar cualquier agente o grupo de agentes de la plataforma de manera que controla el intercambio de mensajes entre dichos agentes. Cada vez que alguno de los agentes seleccionados por el agente *Sniffer* recibe o envía algún mensaje dicho mensaje aparece en la pantalla a modo de flecha con origen en el agente que envía el mensaje y destino en el agente destinatario. Dicha flecha se puede pulsar y así se ve con detalle todas las características del mensaje capturado.

En la parte inferior de la interfaz representada en la figura 3.5 se dispone de dos secciones. En el de la izquierda se muestra el árbol de plataformas y en la parte derecha se puede ver el intercambio de mensajes entre los agentes que se seleccionen.

Las opciones del menú *Actions* más importantes, que van a permitir realizar la depuración de la plataforma multiagente que se este desarrollando, son las siguientes:

- *Do sniff this agent(s):* añade el agente o agentes seleccionados a la sección de la derecha donde se visualiza el intercambio de mensajes.
- *Do not sniff this agent(s):* elimina el agente o los agentes seleccionados de la sección de la derecha donde se visualiza el intercambio de mensajes.
- *Show only agent(s):* añade el o los agente seleccionados de la sección donde se visualiza el intercambio de mensajes, pero solo mostrará el agente, no los mensajes que lleguen hasta él o provengan de él.



- *Clear Canvas*: borra todos los mensajes que estuvieran dibujados hasta el momento en la sección. Se continúa con el proceso de depuración.

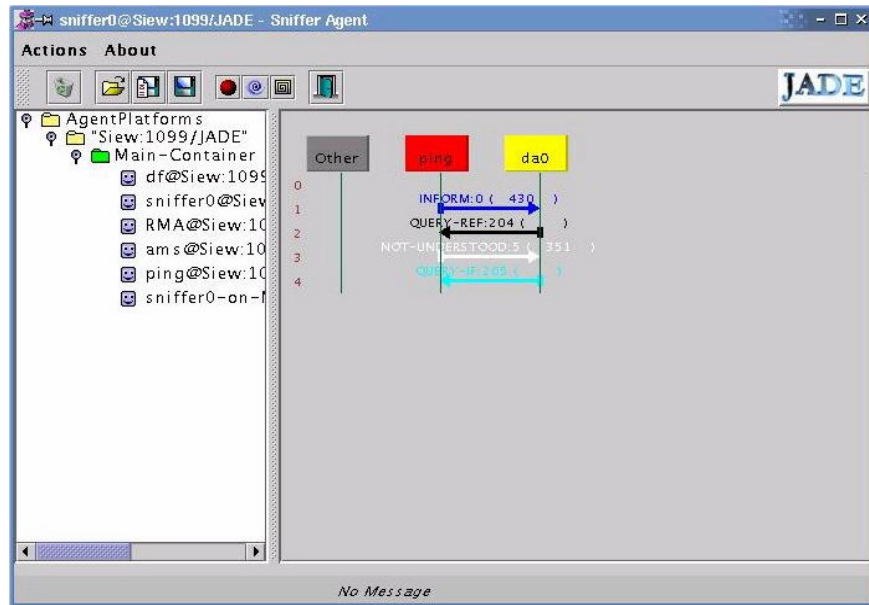


Figura 3.5 Interfaz gráfica del Agente Sniffer

*DF GUI*: La GUI del agente DF (*Directory Facilitator*, visto en el capítulo I) actúa de forma parecida a un agente de páginas amarillas. Se puede usar esta GUI para ver la descripción de los agentes registrados, modificar la descripción de dichos agentes y buscar alguna descripción concreta entre los distintos agentes.

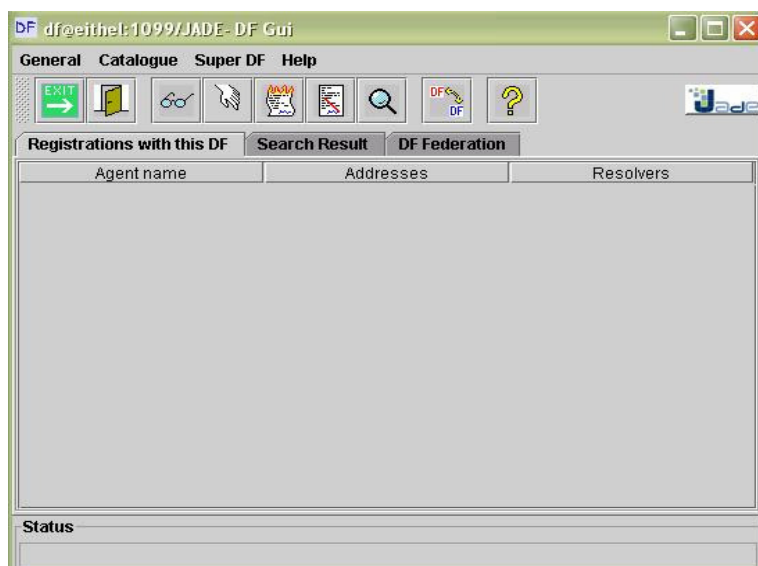


Figura 3.6 Interfaz gráfica del DF GUI

*Agente Introspector*: Permite monitorear y controlar el ciclo de vida de un agente. Muestra las colas de entrada y salida de mensajes

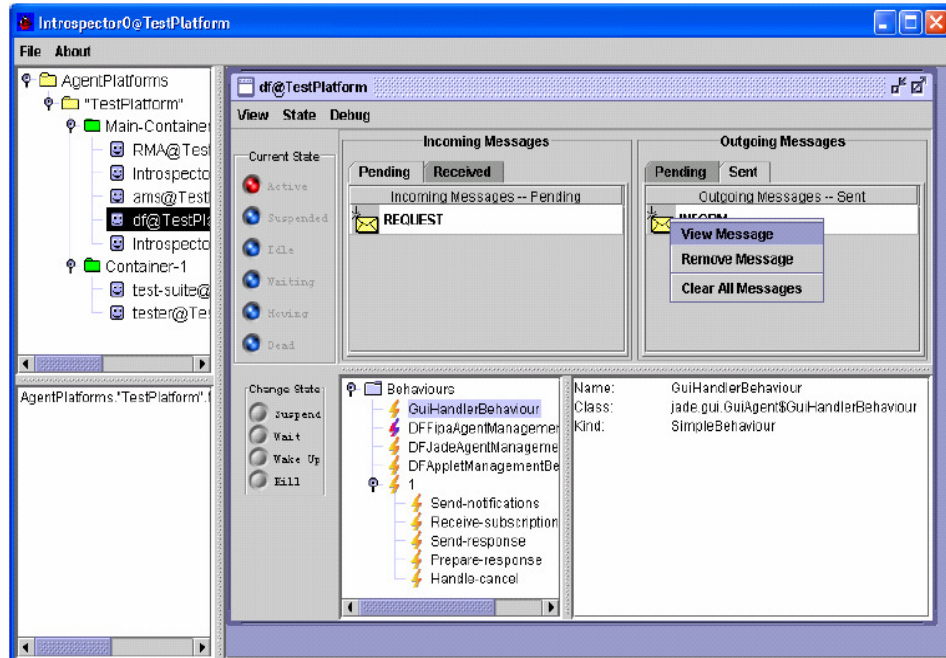


Figura 3.7 Interfaz gráfica del Agente Introspector

### 3.4.- PROGRAMACIÓN DE AGENTES JADE

El kit de desarrollo proporcionado por JADE cuenta con las clases necesarias para la creación de agentes. La clase **jade.core** implementa el kernel del sistema.<sup>44</sup>

- Incluye class Agent para ser extendida por los programadores en sus desarrollos.
- jade.core.behaviour que implementa las tareas e intenciones de los agentes.
- jade.lang.acl: Este paquete tiene por objetivo procesar el lenguaje de Comunicación entre Agentes (ACL). Se tiene los métodos: agent.send () y agent.receive ()).

<sup>44</sup> Tutorial Básico de JADE: galahad.plg.inf.uc3m.es/~docweb/swagent/jade\_texto.pdf

### 3.4.1.- CREACIÓN DE AGENTES

Un agente simplemente es una clase que hereda de la clase `jade.core.Agent` y que como mínimo implemente el método `setup()`. Ejemplo

```
import jade.core.Agent;  
  
public class HelloAgent extends Agent  
{  
    protected void setup()  
    {  
        System.out.println("Hola Mundo. ");  
        System.out.println("Mi nombre es "+ getLocalName());  
    }  
}
```

El método `setup()` se ejecuta cuando se va a lanzar el agente para su ejecución y únicamente debe contener código relativo a tareas de inicialización. Por lo tanto, de momento, el agente no va a hacer absolutamente nada más que emitir ese mensaje cuando se lance.

### 3.4.2.- COMPILACIÓN Y EJECUCIÓN DE AGENTES

Una vez creado el agente, hay que compilarlo mediante el comando:

```
javac HelloAgent.java
```

Una vez compilado se lo ejecuta mediante el comando:

```
java jade.Boot nombre1:clase1 nombre2:clase2 ...
```

Para cada agente se debe indicar su nombre y la clase a la que instancia (se recuerda que cada agente será la instancia de una determinada clase) separado por dos puntos. En este caso sería:

```
java jade.Boot agente1:HelloAgent
```

Si además se quisiera mostrar la interfaz del agente RMA para poder depurar el sistema multiagente, se debe teclear lo siguiente:

java jade.Boot -gui agente1:HelloAgent

Después de haber ejecutado el agente, aparece seguidamente la indicación de que el contenedor principal está listo. Aunque el agente no va a hacer nada más, permanece ejecutándose hasta que se invoca el método `doDelete()`. Este método llama a su vez a `takeDown()` que se puede redefinir para incluir código relativo a limpieza de elementos del agente.

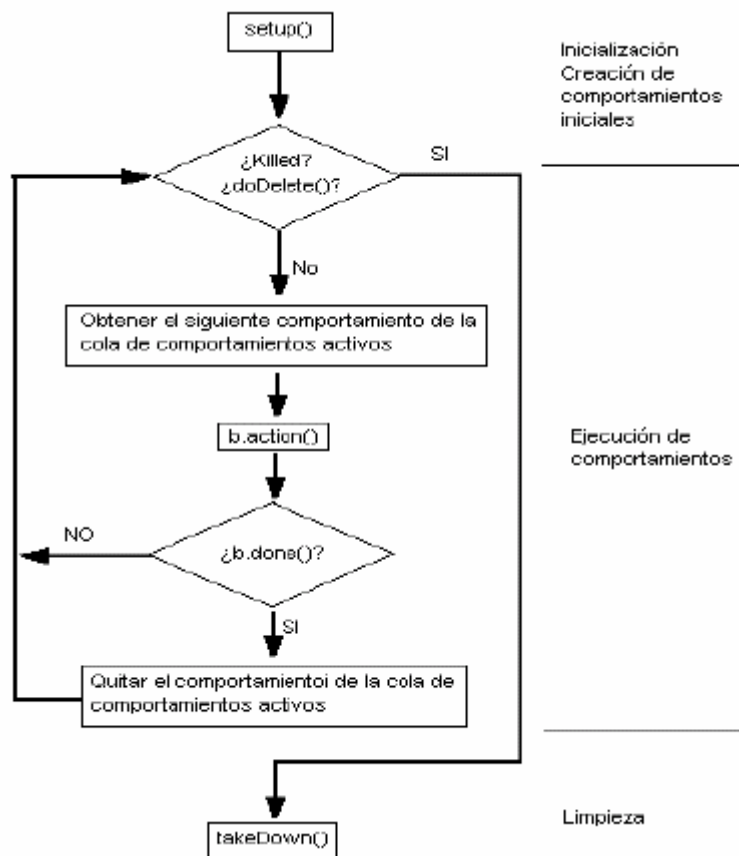
### 3.4.3.- COMPORTAMIENTOS

Un comportamiento es un conjunto de acciones que debe tomar el agente para lograr su objetivo. Los comportamientos se implementan como un objeto de la clase `jade.core.behaviours.Behaviour`.

Los objetos `Behaviour` describen pequeñas tareas que debe realizar el agente ejecutándose según un planificador que se encuentra implementado en la clase *Agent*. El planificador va ejecutando según una política round-robin (o por turnos rotatorios) los objetos `behaviour` que se encuentran en una cola FIFO, existiendo dos métodos, `addBehaviour(behaviourObject)` y `removeBehaviour(behaviourObject)`, que permiten gestionar la entrada y salida de los objetos `behaviour` en la cola del planificador.

Los objetos `behaviour` tienen dos métodos que se deben reescribir, `action()` y `done()`. El método `action()` es en donde se deben desarrollar las tareas que debe realizar el agente, mientras que el método `done()` es llamado cuando `action()` finaliza. El planificador va ejecutando uno a uno los métodos `action()` de la cola y cuando el método `action()` de un objeto finaliza se llama al método `done()`, este debe retornar un booleano, si retorna `true`, el objeto es sacado fuera del planificador ya que se da por concluida su tarea, mientras que si retorna `false` se vuelve a planificar.

Si en algún momento de la ejecución del método `action()` se requiere esperar por la llegada de un mensaje, existe el método `block()` para mandar a una cola de bloqueados cuando el método `action()` finaliza (no cuando se llama a `block()`). (Figura 3.8)



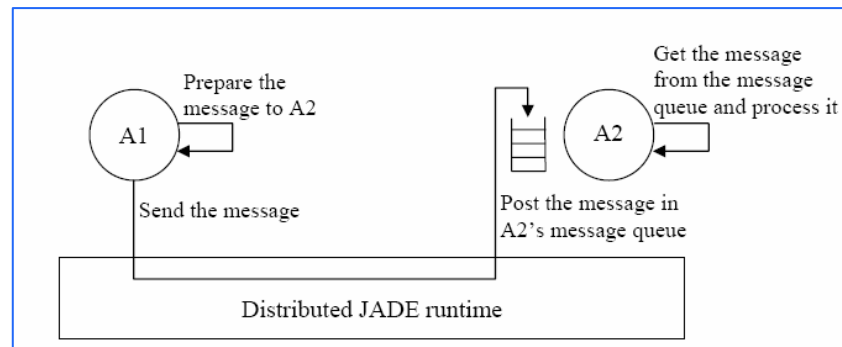
**Figura 3.8 Flujo de control de un agente básico: Inicialización, realización de la tarea y limpieza y finalización.**

Cuando se produce la llegada de un mensaje, todos los objetos en la cola de bloqueados se planifican y deben comprobar si el mensaje llegado es para ellos o no. En caso de que un objeto no sea el destinatario del mensaje debe volver a bloquearse.

Se puede pensar que los behaviours son como los hilos de ejecución JAVA. Igual que las threads en Java, en un agente pueden ejecutarse a la vez tantos comportamientos como sea necesario. Sin embargo, a diferencia de las threads en JAVA, el decidir que comportamiento se ejecuta en cada momento es tarea nuestra (en JAVA lo decide la máquina virtual). Esto demuestra que cada agente equivale a un único thread, lo que permite el ahorro de ciclos de CPU y memoria que esto implica.

### 3.4.4.- COMUNICACIÓN ENTRE AGENTES

JADE sigue el estándar FIPA. Como tal es capaz de hacer que mensajes del tipo ACL puedan intercambiarse entre los agentes implicados, a través del AMS (*Agent Management System*) y mediante un protocolo estándar como, por ejemplo, IIOP (*Internet Inter-ORB Protocol*).



**Figura 3.9 Esquema de envío de mensajes en JADE**

El esquema de envío de mensajes en JADE responde al de la figura 3.9. Como puede verse, aparece una cola de mensajes por cada agente. Cuando un mensaje llega al agente, solamente tiene que mirar en su cola de mensajes y extraerlo de ahí. Para mandar mensajes en JADE, teniendo en cuenta como están definidos los mismos en FIPA, se debe hacer uso de la clase `ACLMessage` como en el ejemplo siguiente:

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("Peter", AID.ISLOCALNAME));
msg.setLanguage("English");
msg.setOntology("Weather-forecast-ontology");
msg.setContent("Hoy día esta lloviendo");
send(msg);
```

en el que, como puede verse, el acto comunicativo es del tipo `inform` (utilizado para el paso de información, visto en el capítulo 1), se envía al agente Peter, en lenguaje de contenido del mensaje es inglés natural, la ontología respectiva es `Weather-forecast-ontology`, y el mensaje es el que aparece en la penúltima línea de código. Para mandar el mensaje se hace

uso del método `send()` de la clase `Agent`. (Revisar la documentación de JADE <http://jade.tilab.com>)

### 3.4.5.- ONTOLOGÍA

La finalidad de la ontología dentro de los Sistemas Multiagentes (*SMA*) es facilitar en la medida de lo posible la comunicación y compartición de datos entre sistemas independientes mediante la creación de esquemas que contienen las entidades relevantes así como sus relaciones dentro del dominio.<sup>45</sup>

En JADE, la definición de ontologías se implementa definiendo clases que representen el conocimiento con el que se va a trabajar, las cuales hereden de clases predefinidas por jade que pueden codificar y decodificar mensajes en un formato FIPA estándar, permitiendo así la interoperabilidad con otros sistemas de agentes. Para ello proporciona un soporte para ontologías que permite realizar la transformación entre ACL Message y un objeto Java de fácil manipulación.

La transformación se hace a través de una ontología, incluida en el paquete *jade.content.onto*, y a través de unos codecs del contenido del lenguaje, incluidos en el paquete *jade.content.lang*. Por un lado la ontología realiza la validación semántica del mensaje y los codecs realizan la traducción a un string según las reglas sintácticas del lenguaje.

#### 3.4.5.1.- Conversión realizada por el soporte de JADE para ontologías.

Jade incorpora, en el paquete *jade.content*, soporte (codecs), para dos lenguajes de contenido:

- El lenguaje SL es legible por los humanos y codifica las expresiones como string.
- El lenguaje LEAP no es legible por los humanos y es byte-encoded.

---

<sup>45</sup> Tutorial de Agentes: <http://programacionjade.wikispaces.com/Ontolog%C3%ADas>

Una ontología es una instancia de la clase *jade.content.onto.Ontology* en la cual se definen los *Schemas*, conjuntos de elementos que definen la estructura de los predicados, las acciones de los agentes y conceptos relevantes al dominio del problema.

- Predicados: expresiones sobre el estado del mundo. Se utilizan típicamente en mensajes INFORM y QUERY-IF, no en REQUEST. (Ver capítulo 1)
- Acciones de los agentes: expresiones que indican acciones que pueden realizar los agentes. Típicamente se utilizan en mensajes de tipo REQUEST.
- Conceptos: expresiones que representan objetos, representan una estructura con varios atributos. No aparecen aislados en los mensajes sino incluidos en otros elementos.

A continuación se presenta un ejemplo de una ontología para una tienda de frutas, que se puede utilizar para enviar ofertas entre agentes. Primero se identifican los diferentes tipos de esquemas de una ontología:

- Conceptos: representan las entidades que forman parte de la ontología, en este caso se tendrá un concepto llamado “Frutas”.
- Predicados: son expresiones que relacionan a los conceptos para decir algo. Son necesarios, ya que en un mensaje nunca se podrá enviar conceptos, sino predicados o acciones. En este ejemplo “Oferta” es un predicado que indicará que un agente oferta una determinada fruta.
- Acciones: son acciones que pueden llevar a cabo los agentes. Para la ontología de frutas una acción será “Comprar” que indicará a un agente que debe comprar una determinada fruta.

#### **3.4.5.2.- Uso de la herramienta Protégé y BeanGenerator para la creación de ontologías**

Para facilitar la realización de la ontología existe un editor de ontologías open source de libre distribución llamado Protégé, el cual es una herramienta que puede trabajar con muchos plugins como Eclipse. Gracias a un plugin llamado Beangenerator es posible definir ontologías usando Protégé y dejando a BeanGenerator el trabajo de crear el código



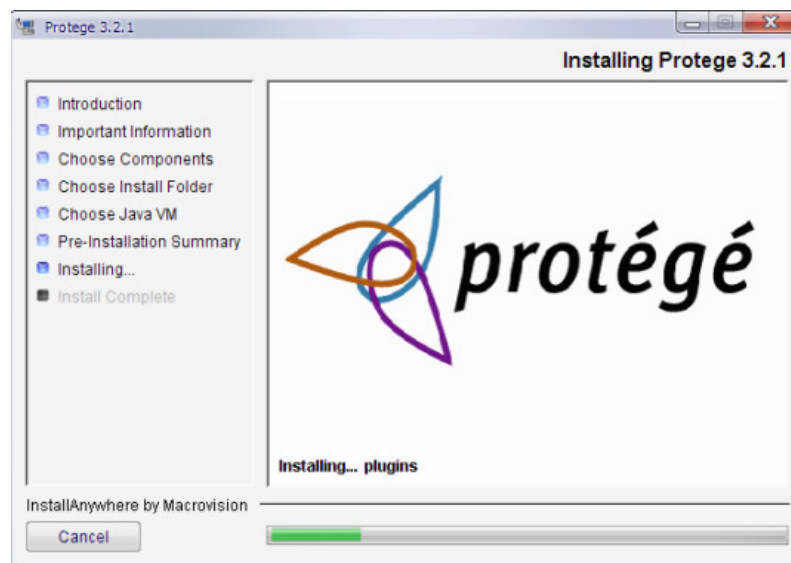
fuente. Esto permite trabajar con una interfaz gráfica a la hora de definir las ontologías en vez de tener que escribir el código fuente para las clases.

Se puede descargar Protégé de forma gratuita desde la dirección <http://protege.stanford.edu/>. Y el plugin BeanGenerator desde la dirección <http://protege.cim3.net/cgi-bin/wiki.pl?OntologyBeanGenerator>.

Una vez descargados estos archivos se procede a la instalación de *Protégé* para la cual se necesita una máquina virtual de Java instalada en el sistema.

Terminado este proceso se descomprime el archivo de *Beangenerator* en la carpeta plugins dentro del directorio donde se ha instalado Protégé..

El desarrollo de la ontología perteneciente al sistema del presente proyecto se encuentra detallado en el Capítulo 4.



**Figura 3.10** Interfaz gráfica de instalación de la herramienta Protégé

Para más información de cómo crear una ontología revisar el sitio Web: <http://programacionjade.wikispaces.com/Ontolog%C3%ADas>

# **CAPÍTULO IV**

## **IV.- DESARROLLO DEL SOFTWARE**

### **4.1.- INTRODUCCIÓN**

Para resolver el problema de la construcción de un sistema software, se debe cumplir con un proceso, el cual está formado por una colección de actividades que comienza con la identificación de una necesidad y concluye con el retiro del software que satisface dicha necesidad.

El proceso de desarrollo de software más básico está formado por cinco etapas: análisis, diseño, implementación, pruebas e implantación. En el análisis se estudia el problema y se termina con una especificación completa del comportamiento externo que debería tener el sistema a construir. En el diseño se descompone el sistema software en sus componentes principales, y éstos se dividen a su vez en componentes más pequeños, además se definen y documentan los algoritmos que llevarán a cabo la función a realizar por cada módulo. En la implementación se transforman los algoritmos definidos en el diseño en un lenguaje comprensible para una computadora. En la etapa de pruebas se realiza el proceso de comprobación para eliminar los errores. Y en la etapa de implantación se instala el sistema software.

Durante este proceso se selecciona un ciclo de vida que establece el orden de ejecución de las distintas actividades, marcadas por el proceso e involucradas en el proyecto.

La metodología designada a usarse en este proyecto es INGENIAS y esta enfocada a un ciclo de vida iterativo e incremental, lo cual permitirá que en cualquier momento el desarrollador pueda realizar cambios que considere necesarios para resolver las necesidades del cliente o usuario.

## **4.2.- ANÁLISIS**

### **4.2.1.- ESPECIFICACIÓN DE REQUISITOS**

#### **4.2.1.1.- Introducción**

El presente documento permite detallar la Especificación de Requisitos de Software (ERS) para el desarrollo de la AGENDA MULTIAGENTE ORIENTADA A LA WEB PARA LA GESTIÓN DE REUNIONES UTILIZANDO LA PLATAFORMA JADE – JAVA EN LA ESCUELA POLITÉCNICA DEL EJÉRCITO SEDE LATACUNGA. Esta especificación se ha estructurado basándose en las directrices dadas por el estándar “IEEE Recommended Practice for Software Requirements Specification ANSI/IEEE 830”.

##### **4.2.1.1.1.- Propósito**

El propósito que presenta este documento es el de mostrar en forma clara y concisa los requerimientos que debe satisfacer o cumplir el software a desarrollarse, define también las funcionalidades y restricciones del sistema con el fin de facilitar la gestión de reuniones entre el personal docente de la Escuela Politécnica del Ejército sede Latacunga (ESPE-L).

##### **4.2.1.1.2.- Ámbito del Sistema**

El motor que impulsa al desarrollo de este sistema es la necesidad de convocar a reuniones entre los docentes de todos los estamentos la ESPE-L en una fecha y hora disponible para cada uno de ellos, tomando en consideración su agenda personal. Por tal motivo se desarrollará un software que administre estas gestiones y sirva de ayuda para el desenvolvimiento del personal mencionado en sus diferentes funciones.

Este sistema se lo ha denominado AMulGeR (*Agenda Multiagente para la Gestión de Reuniones*) y está enfocado a controlar tres gestiones, las cuales son: Administrativa, Actividades y Reuniones.

#### 4.2.1.1.3.- *Definiciones, Acrónimos y Abreviaturas.*

##### 4.2.1.1.3.1.- *Definiciones*

Administrador	Persona encargada de actualizar la información desde el sistema escolástico de la ESPE-L al sistema AMulGeR.
Usuario	Persona encargada de manejar la agenda y gestionar reuniones

**Tabla 4.1. Definición de términos**

##### 4.2.1.1.3.2.- *Acrónimos*

ERS	Especificación de requisitos de Software
BD	Base de Datos
MySQL	My Structured Query Language
JADE	Java Agents Development Environment
JDK	Java Development Kit

**Tabla 4.2. Acrónimos**

##### 4.2.1.1.3.3.- *Abreviaturas*

AMulGeR	<b>Agenda Multiagente para la Gestión de Reuniones</b>
---------	--

**Tabla 4.3. Abreviaturas**

##### 4.2.1.1.4.- *Referencias*

Estándar IEEE Recommended Practice for Software Requirements Specification ANSI/IEEE 830 para la especificación de requisitos de Software que constituye este documento.

#### ***4.2.1.1.5.- Visión General del Documento***

Este documento consta de tres secciones, esta sección es la introducción y proporciona una visión general de ERS. En la segunda sección se da una descripción general del sistema, con el fin de conocer las principales funciones que debe realizar, los datos asociados y los factores, restricciones y dependencias que afectan al desarrollo, y en la tercera sección se detallan los requisitos que debe satisfacer el sistema. Además se presenta formularios del sistema en la sección de anexos de éste documento.

#### **4.2.1.2.- Descripción General**

Esta sección presenta una descripción general del sistema AMulGeR, con el fin de dar a conocer el funcionamiento de las tres gestiones que abarcará el software juntamente con los datos a manejar, restricciones y cualquier factor que pueda afectar su desarrollo.

##### ***4.2.1.2.1.- Funciones del Sistema***

En términos generales el sistema deberá proporcionar soporte a las siguientes gestiones:

- Gestión Administrativa
- Gestión de Actividades
- Gestión de Reuniones

##### ***4.2.1.2.1.1.- Gestión Administrativa***

La ESPE-L dispone actualmente de un sistema escolástico, cuya base de datos (BD) utilizada es Sybase. Esta BD contiene información necesaria para el sistema AMulGeR como son: docentes, facultad, periodo académico, día, paralelo, materia, horarios. Por lo tanto esta gestión permitirá replicar la información mencionada anteriormente del sistema escolástico a la base de datos en MySQL que el sistema AMulGeR utilizará.

Para efectuar esta actualización el Administrador seleccionará el código del periodo académico actual y el sistema procederá a replicar la información correspondiente a ese periodo.

El Administrador también podrá limpiar la base de datos del sistema AMulGeR cuando haya terminado el periodo académico, se realizará esta acción cuando existan datos en la BD del sistema AMulGeR.

Cabe recalcar que cuando se realice alguna modificación de la base de datos del sistema escolástico dentro del periodo académico, el Administrador deberá nuevamente replicar la información al sistema AMulGeR.

#### ***4.2.1.2.1.2.- Gestión de Actividades***

En esta gestión se realizará las funciones de altas, bajas, cambios, consulta individual y general de actividades.

Para dar de alta una actividad, el usuario seleccionará la opción que le permita realizar esta función, digitará el asunto, la ubicación, la descripción y seleccionará el lugar (dentro o fuera de la ESPE-L), el estado (no comenzada, completada, aplazada), la hora de inicio y la hora de finalización de la actividad. Cuando el usuario seleccione la opción grabar, el sistema verificará si todos los campos han sido llenados y si los datos ingresados son correctos.

Para la baja de una actividad, el usuario deberá seleccionar una de ellas de la consulta general presentada, a continuación podrá visualizar individualmente la actividad y elegirá la opción eliminar. El sistema preguntará antes de ejecutar esta acción si esta seguro de hacerlo.

Para modificar una actividad, el usuario deberá consultarlas de forma general y seleccionar una de ellas. A continuación ingresará la nueva información en los campos a

modificar. Cuando el usuario seleccione la opción modificar, el sistema verificará si todos los campos han sido llenados y si los datos ingresados son correctos.

Para consultar de forma general las actividades por día, el usuario deberá seleccionar una fecha de un calendario. Podrá visualizar solamente el asunto, ubicación, hora inicio y hora fin de las actividades. En esta operación también se visualizará la facultad, materia, aula, día, hora inicio y hora fin correspondiente a los horarios de clases del docente de la respectiva fecha seleccionada.

Asimismo, para poder consultar individualmente todos los datos de una actividad, deberá seleccionar una de ellas de la consulta general presentada.

#### ***4.2.1.2.1.3.- Gestión de Reuniones***

Esta gestión permitirá convocar, modificar el lugar, cancelar la participación de un invitado, cancelar una reunión como también consultarlas en forma general e individual.

Para convocar una reunión el usuario ingresará la siguiente información: descripción, lugar, tiempo de duración, número mínimo de participantes, seleccionará la fecha mínima y máxima entre las cuales desea que se convoque la reunión (en prioridad baja), la fecha y hora en las que se desea realizar la reunión (en prioridad alta), los docentes invitados o participantes (mínimo dos, incluido el que convoca), que pueden ser seleccionados de una lista que se podrá filtrar dependiendo de la facultad que pertenecen, y la prioridad (alta o baja). Éste último campo consiste en que si se elige la prioridad alta no se considerará ninguna actividad del convocante y de cada participante, es decir, se establecerá la reunión en la fecha y hora designada por el usuario convocante. En cambio en la prioridad baja, se procederá a buscar la fecha común de las horas libres disponibles de todos los participantes. Si todos los campos han sido llenados y los datos son correctos el sistema los aceptará y procederá a convocar la reunión.

El sistema deberá buscar las fechas y horas libres de los invitados a la reunión, incluido el usuario que la convoca. Seguidamente el sistema obtendrá una fecha y hora

común de entre todas las fechas y horas libres encontradas para realizar la reunión y deberá obtener por lo menos el número mínimo. Una vez terminado este proceso se almacenará la reunión en la agenda de cada uno de los participantes y se informará a cada uno de ellos por medio de un mensaje dentro de su agenda.

En la prioridad alta se presenta el caso de prioridades entre docentes. Cuando uno de ellos de mayor rango convoque una reunión en un intervalo de hora que interfiere con otra hora de una reunión ya establecida que convocó un docente de menor rango, ésta será eliminada y se informará a los implicados en las reuniones de la gestión realizada.

A continuación se menciona los casos en los que no se podrá convocar la reunión:

- El usuario convocante no dispone de una fecha y horas libres
- Si no hubiera suficientes invitados para convocar la reunión debido a que el número mínimo requerido no dispone de fechas y horas libres.
- Si no existiera una fecha común de los participantes o del número mínimo.

Cuando un usuario ingresa a esta gestión podrá visualizar la siguiente información: lugar, descripción, fecha y hora de inicio de todas las reuniones en las que él es participante, leído y motivo. Al seleccionar una de ellas podrá consultar individualmente todos los campos además del nombre del docente que convocó la reunión.

Se permitirá modificar solamente el lugar de reunión, para ello el usuario convocante, seleccionará la reunión a cambiar de la consulta general e ingresará la nueva información. Al escoger la opción editar el sistema procede a modificar el lugar en la agenda del convocante y de los invitados, además de informarles el cambio realizado por medio de un mensaje dentro de sus agendas.

Un usuario no convocante puede solicitar cancelar su participación en la reunión, para ello deberá seleccionar la reunión de la consulta general y escoger la opción no asistir. El sistema verificará si existe aun invitados suficientes para realizar la reunión, si es así, procede a modificar los invitados en cada agenda, además de informarles el cambio



realizado. En caso de que no exista el número mínimo de participantes se eliminará o cancelará la convocatoria a la reunión.

Para que un usuario convocante pueda eliminar o cancelar una reunión, debe seleccionar una de ellas de la consulta general y escoger la opción eliminar. En seguida el sistema preguntará antes de ejecutar esta acción si está seguro de hacerlo y procederá a eliminar la reunión de la agenda del usuario convocante y de los invitados e informará al usuario y los participantes que se canceló una reunión por medio de un mensaje dentro de sus respectivas agendas.

#### **4.2.1.2.2.- Características de los usuarios**

Este sistema proporcionará una interfaz de usuario fácil de aprender y manejar, además presentará un alto grado de usabilidad, permitiendo que el usuario navegue sin ninguna dificultad.

#### **4.2.1.2.3.- Restricciones**

Para que el sistema funcione correctamente se requerirá de una red con plataforma cliente / servidor.

#### **4.2.1.2.4.- Dependencias**

El sistema a desarrollar depende de la base de datos del sistema escolático de la ESPE-L y del Portal Académico en la gestión de actividades y reuniones para su correcto funcionamiento.

#### **4.2.1.3.- Requisitos Específicos**

En este apartado se presentan los requisitos funcionales que deberán ser satisfechos por el sistema. Todos los requisitos aquí expuestos son *esenciales*, es decir, no sería aceptable un sistema que no satisfaga alguno de los requisitos aquí presentados.

#### **4.2.1.3.1.- Requisitos Funcionales**

##### **4.2.1.3.1.1.- Gestión Administrativa**

- Req. (01) Replicar base de datos
- Req. (02) Limpiar base de datos

##### **4.2.1.3.1.2.- Gestión de Actividades**

- Req. (03) Dar de alta una actividad
- Req. (04) Dar de baja una actividad
- Req. (05) Modificar una actividad
- Req. (06) Consultar de forma general las actividades y horarios
- Req. (07) Consultar de forma individual las actividades

##### **4.2.1.3.1.3.- Gestión de Reuniones**

- Req. (08) Convocar reunión
- Req. (09) Cancelar una reunión
- Req. (10) Cambiar el lugar de la reunión
- Req. (11) Cancelar la participación de un invitado
- Req. (12) Informar del estado de las reuniones
- Req. (13) Consultar de forma general las reuniones
- Req. (14) Consultar de forma individual las reuniones

#### **4.2.1.3.2.- Requisitos de Interfaces Externas**

##### **4.2.1.3.2.1.- Interfaces de Usuario**

La interfaz de usuario se regirá bajo los estándares utilizados en los sistemas de la ESPE-L, la cual es orientada al manejo de páginas web mediante hipervínculos y ventanas. Se podrá manipular el sistema mediante el teclado y el ratón.

#### ***4.2.1.3.2.2.- Interfaces Hardware***

El sistema se basará en una plataforma cliente / servidor que tiene a disposición la ESPE-L.

#### ***4.2.1.3.2.3.- Interfaces Software***

Las gestiones a realizar podrán ser incluidas como un módulo más al Portal de la ESPE-L (para la gestión de reuniones se necesita obligatoriamente los requerimientos de software mencionados más adelante), por lo que, para poder ingresar al sistema AMulGeR se deberá navegar primero por las páginas existentes en este portal.

#### ***4.2.1.3.2.4.- Interfaces de Comunicación***

La conexión se establecerá mediante una Intranet y consecutivamente a Internet. Esto será transparente para la aplicación

#### ***4.2.1.3.3.- Requisitos de Rendimiento***

El rendimiento del sistema dependerá en gran parte de la velocidad del procesador del Servidor Web, base de datos y del Proveedor de Internet

#### ***4.2.1.3.4.- Requisitos de Desarrollo***

El ciclo de vida elegido para desarrollar el producto será el de prototipo iterativo incremental, de manera que se puedan incorporar fácilmente cambios y nuevas funciones.

#### ***4.2.1.3.5.- Requisitos Tecnológicos***

Los requisitos mínimos para que el software funcione correctamente son:

Para el Servidor

- Procesador: Pentium IV

- Memoria: 512 Mb
- Espacio libre en disco: 10 Gb
- Tarjeta Ethernet

Para el Cliente

- Procesador: Pentium IV
- Memoria: 256 Mb
- Espacio libre en disco: 200 Mb
- Tarjeta Ethernet

#### **4.2.1.3.6.- Requisitos Software**

- La aplicación necesita tres Servidores Web: Internet Information Server, Apache y Apache Tomcat para ejecutar la gestión administrativa, de actividades y reuniones respectivamente. Esto se debe a que el sistema AMulGeR debe adecuarse a los lenguajes de programación (Microsoft Visual Studio 2005, lenguaje C#, Framework .NET 1.0 y PHP) con sus respectivos Servidores Web utilizados en la ESPE-L y necesita del lenguaje JAVA para la programación de los agentes bajo la plataforma JADE en la gestión de reuniones.
- La base de Datos a utilizar es MySQL.
- Se debe disponer del JDK (*Java Development Kit*) 4.0 como mínimo para que la gestión de reuniones pueda desarrollarse y ejecutarse.
- Se requiere de las librerías de la plataforma JADE 6.0 que se puede descargar del sitio web: <http://jade.tilab.com>.
- El sistema Operativo a utilizar para la gestión administrativa y de actividades debe ser Windows y para la gestión de reuniones Windows o Linux.

#### **4.2.1.3.7.- Atributos**

##### **4.2.1.3.7.1.- Seguridad**

En el sistema se identificará dos tipos de usuarios: el Administrador y Usuario. El primero tendrá acceso a la Gestión Administrativa y el segundo a la gestión de

Actividades y Reuniones. Cabe recalcar que la parte de autenticación de usuarios (login y password) se encuentra ya desarrollada en el Portal Académico de la ESPE-L, por lo que el sistema AMulGeR deberá capturar el resultado de aquella autenticación para poder mostrar la agenda con los datos respectivos al usuario que ingresó al sistema.

#### 4.2.1.4.- Anexo Documentos

##### 4.2.1.4.1.- Formularios de Entrada

<input type="text"/>	Ingreso
<input type="text"/>	Selección

Asunto	<input type="text"/>	Lugar	<input type="text"/>
Ubicación	<input type="text"/>	Hora de inicio	<input type="text"/>
Fecha de la actividad	<input type="text"/>	Hora de finalización	<input type="text"/>
Estado	<input type="text"/>		
Descripción	<input type="text"/>		

**Figura 4.1. Ingreso y/o modificación de una actividad**

Descripción	<input type="text"/>	Prioridad	<input type="text"/>
Lugar	<input type="text"/>	Tiempo de duración	<input type="text"/>
Fecha mínima	<input type="text"/>	Num. Min. participantes	<input type="text"/>
Fecha máxima	<input type="text"/>	Hora reunión	<input type="text"/>
Fecha reunión	<input type="text"/>		
Participantes	<input type="text"/>		

**Figura 4.2. Convocar reunión y/o modificar lugar de una reunión**

4.2.1.4.2.- Formularios de Salida

Fecha		<input style="width: 100px; height: 20px;" type="text"/>			
<b>Horarios</b>					
Facultad	Materia	Aula	Día	Hora inicio	Hora fin
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
<b>Actividades</b>					
	Asunto	Ubicación	Hora inicio	Hora fin	
	_____	_____	_____	_____	
	_____	_____	_____	_____	

Figura 4.3. Consulta general de actividades y horarios

Asunto	_____	Lugar	
Ubicación	_____	Hora de inicio	_____
Fecha de la actividad	_____	Hora de finalización	_____
Estado	_____		
Descripción	_____		

Figura 4.4. Consulta individual de una actividad

<b>Reuniones</b>					
Lugar	Descripción	Fecha	Hora inicio	Leído	Motivo
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____

Figura 4.5. Consulta general de reuniones

Descripción	_____	Fecha	_____
Lugar	_____	Hora de inicio	_____
Invita	_____	Num. Min. participantes	_____
Hora finalización	_____		
Participantes	_____		

**Figura 4.6. Consulta individual de una reunión**

#### **4.2.2.- HERRAMIENTA DE MODELADO DE INGENIAS: IDK (INGENIAS DEVELOPMENT KIT)**

El IDK es una herramienta visual, 100% Java, ofrecida gratuitamente por la Metodología INGENIAS. Permite crear y modificar modelos de SMA similares a los diagramas de UML, genera documentación (HTML), saca snapshots (imágenes) de los diagramas para utilizarlos en otras aplicaciones, procesa las especificaciones mientras se están generando con el editor o una vez grabados en un fichero, etc.

Esta herramienta va a ser utilizada para modelar el sistema AMulGeR, por lo tanto para su instalación se debe seguir los siguientes pasos<sup>46</sup>:

1. Descargar el IDK de INGENIAS desde la Web Oficial de la Metodología:  
<http://grasia.fdi.ucm.es/ingenias/Spain/lenguaje/index.php>.
2. Descomprimir en el directorio: c:\
3. Desde la ventana de comandos, ubicarse en el directorio: c:\IDK y poner:  
java -jar lib\ingeniaseditor.jar. Para poder arrancar el Editor se necesita tener instalado el JDK (*Java Development Kit*) 1.5 o superior

<sup>46</sup> Web Oficial Metodología <http://grasia.fdi.ucm.es/ingenias/Spain/downloads/index.php>

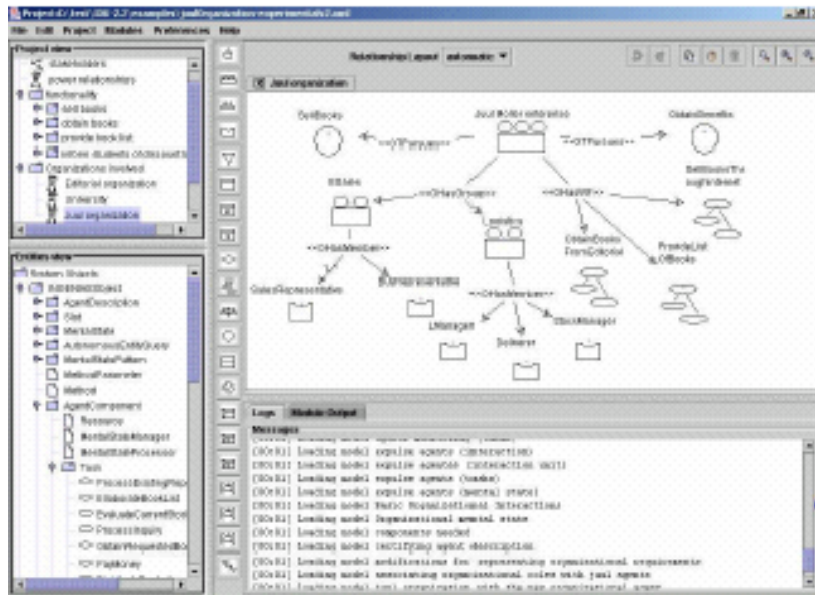


Figura 4.7 Editor INGENIAS IDE

### 4.2.3.- CASOS DE USO

En esta etapa se analizan los casos de uso que se consideran relevantes para obtener la arquitectura del sistema y se asocian con modelos de interacciones para estudiar cómo se llevan a cabo.

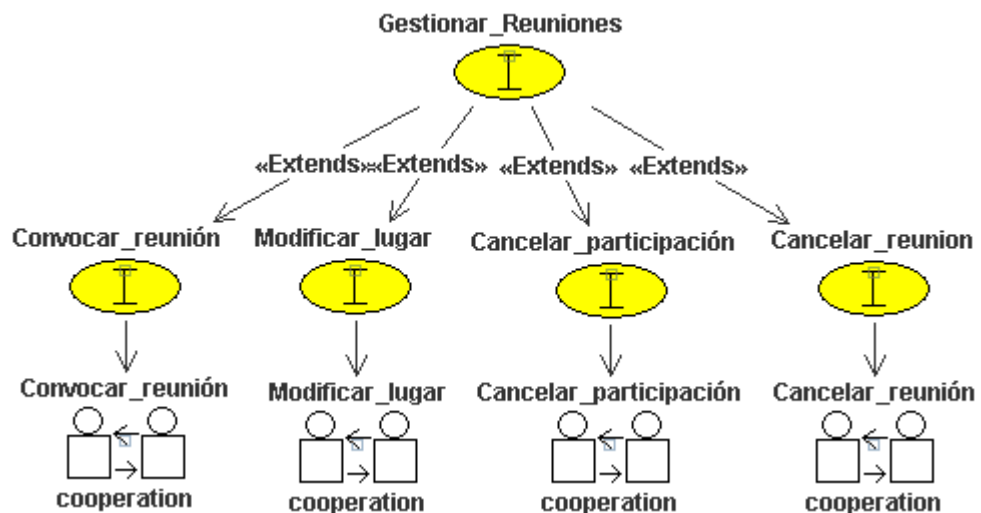


Figura 4.8. Casos de uso asociados



#### 4.2.3.1.- Caso de usos detallados

##### **Caso de uso:** Gestionar\_Reuniones

De este caso de uso se extiende los demás, que a continuación se detallan.

##### **Caso de uso:** Convocar\_reunión

**Actor** Usuario.

##### **Descripción:**

1. El usuario selecciona la operación Convocar Reunión, a continuación el sistema presenta la pantalla para el ingreso de datos y cubrirá los datos necesarios para la convocación de la misma. *[Excepción: Datos incorrectos]*.
2. El gestor de reuniones solicita al gestor de búsquedas las fechas y horas libres (entre las fechas establecidas) de todos los participantes involucrados en la reunión incluida el que la convoca.
3. Una vez que el gestor de reuniones tiene las fechas y horas libres de los participantes en la reunión, verifica si el convocante dispone de horas libres y si existen el número mínimo y se empieza a buscar la fecha y hora común. *[Excepción: El usuario convocante no dispone de fechas y horas libres]* *[Excepción: No hay suficientes invitados para convocar la reunión]* *[Excepción: No hay fecha común]*.
4. Una vez que se convocó la reunión se almacenará en la agenda de los participantes además de informarles.

##### **Excepciones:**

1. *Datos incorrectos*: se puede presentar por la fecha inicial mayor que la fecha final o el número de participantes menor de dos, si no se ha seleccionado ningún invitado o por no ingresar todos los datos.
2. *El usuario convocante no dispone de fechas y horas libres*: esta excepción ocurre cuando no existen fechas ni horas libres del usuario convocante. En este caso el gestor de reuniones informará al usuario que no se puede convocar la reunión, y se termina la operación.

3. *No hay suficientes invitados para convocar la reunión:* si el número de invitados que tienen las fechas libres para convocar la reunión es menor que el número mínimo de participantes en la reunión, se informará al usuario de que no se puede convocar a la misma y se terminará la operación.
4. *No hay fecha común:* Se produce cuando a pesar de que los participantes tengan fechas libres, no existe una fecha común. Se informará al usuario de que no se puede convocar la reunión y se termina la operación.

**Caso de uso:** Modificar\_lugar

**Actor:** Usuario.

**Descripción:**

1. El usuario seleccionará la reunión que quiere modificar [*Excepción: El usuario no es el convocante*].
2. El usuario modificará el lugar, y solicita al gestor de reuniones que realice el cambio en los datos de la reunión. Este último a su vez solicita al gestor agenda que realice esta acción.
3. Una vez que se modificó el lugar se informará a todos los participantes de la reunión

**Excepciones:**

1. *El usuario no es el convocante:* Solo se permitirá modificar una reunión al usuario que convocó la misma

**Caso de uso:** Eliminar\_participante

**Actor:** Usuario.

**Descripción:**

1. El usuario selecciona la reunión en la que quiere cancelar su participación [*Excepción: Es usuario convocante*].
2. Se solicita al gestor de reuniones que elimine al participante en la reunión [*Excepción: No hay participantes suficientes*], y este solicita al gestor agenda para que realice esta acción.

3. El gestor de avisos informará a los usuarios participantes que cierto invitado ha cancelado su participación

**Excepciones:**

1. *El usuario es el convocante:* Un usuario convocante de una reunión no puede cancelar su participación en la misma.
2. *No hay participantes suficientes:* El número de participantes en la reunión debe ser mayor o igual al número mínimo que se necesita para que se realice, caso contrario se procede a desconvocar la reunión (caso de uso cancelar reunión).

**Caso de uso:** Eliminar\_Reunión

**Actor:** Usuario.

**Descripción:**

1. El usuario selecciona la reunión que desea cancelar [*Excepción: El usuario no es el convocante*].
2. El usuario solicita al gestor de reuniones que cancele la reunión elegida y éste a su vez solicita al gestor agenda que elimine la reunión de las agendas de los invitados
3. El gestor de avisos informa a los participantes que la reunión se canceló.

**Excepciones:**

1. *El usuario no es el convocante:* Solo se permitirá cancelar una reunión al usuario que convocó la misma.

**4.2.4.- META-MODELO DE AGENTE**

En el presente meta-modelo se describen los agentes que componen el sistema (en este caso se ha identificado cuatro agentes), sus funcionalidades, sus requisitos, autonomía e inteligencia que son función directa de la especificación del comportamiento del agente (modelos de interacción, organización, tareas y objetivos).

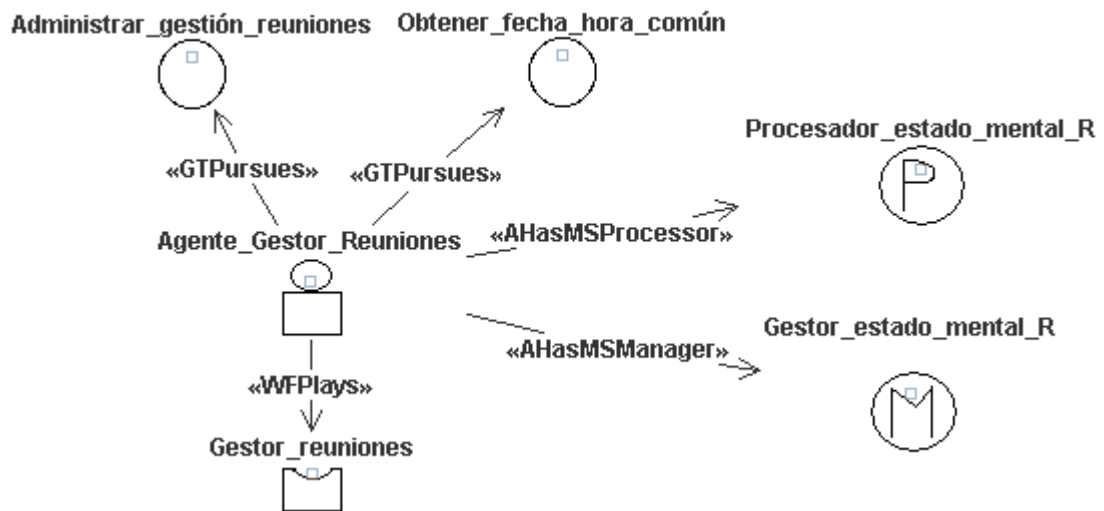


Figura 4.9. Modelo de agente para el Agente Gestor Reuniones

El *Agente\_Gestor\_Reuniones* asume los objetivos de administrar la gestión de reuniones y obtener la fecha y hora común. Estos objetivos se pueden alcanzar gracias a que el agente desempeña un rol llamado *Gestor\_reuniones* que ha sido identificado en las interacciones.

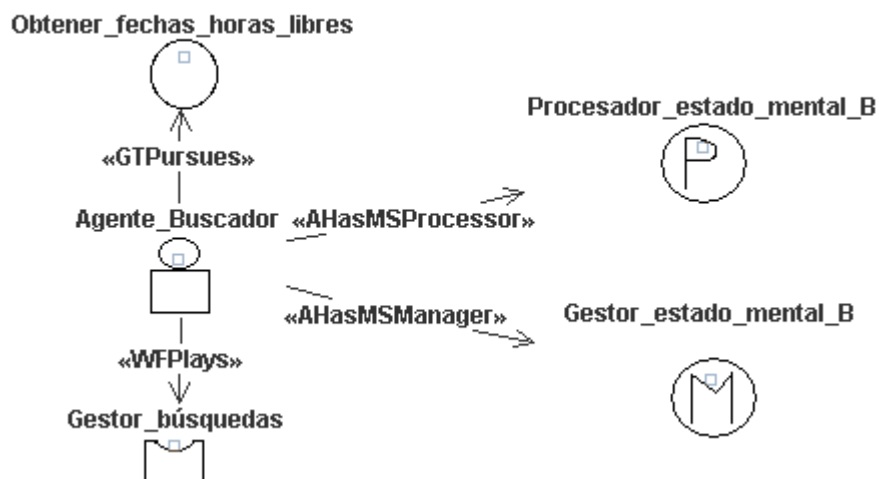


Figura 4.10 Modelo de agente para el Agente Buscador

El *Agente\_Buscador* tiene como objetivo, obtener las fechas y horas libres de un invitado a una reunión. Además desempeña un rol llamado *Gestor\_búsquedas* que tiene la responsabilidad de realizar ciertas tareas para alcanzar el objetivo (ver meta-modelo de objetivos y tareas).

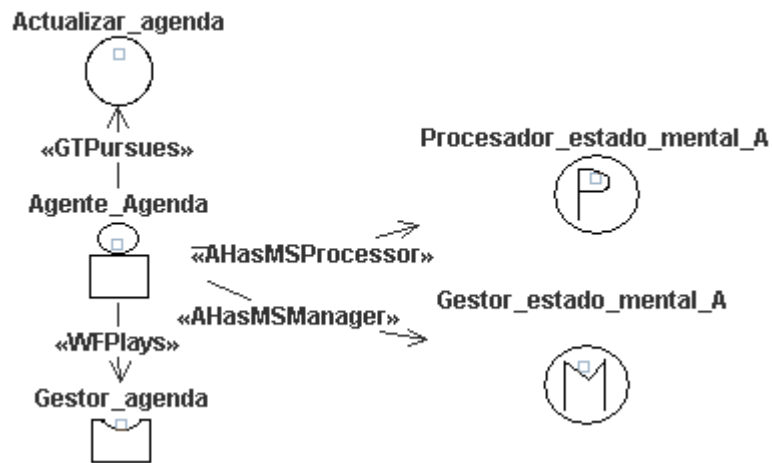


Figura 4.11 Modelo de agente para el Agente Agenda

El *Agente\_Agenda* tiene como objetivo actualizar la agenda ante cualquier operación que realice el usuario por medio del rol *Gestor\_agenda*.

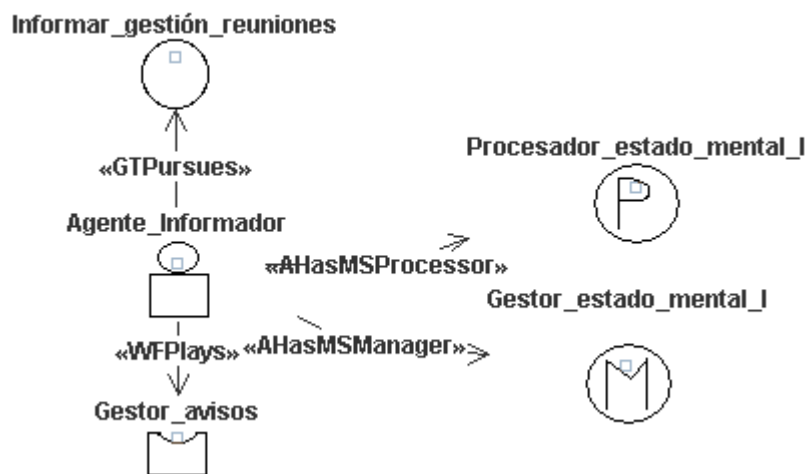


Figura 4.12 Modelo de agente para el Agente Informador

El *Agente\_Informador* asume la responsabilidad de Informar la gestión de reuniones a los participantes de una reunión por medio del rol *Gestor\_avisos* que desempeña este agente.

Los procesadores de estado mental (las decisiones) de los modelos de agentes anteriores se producen de forma algorítmica, en función a las tareas a ejecutar y el beneficio a obtener, u obedecer a la consecución de objetivos eligiendo secuencia de tareas a ejecutar. Los gestores de estado mental se van formando por la relaciones entre las tareas, relaciones entre el agente y objetivos.

#### 4.2.5.- META-MODELO DE INTERACCIONES

Para detallar los casos de uso, se utilizan los modelos de interacción que se presentan a continuación.

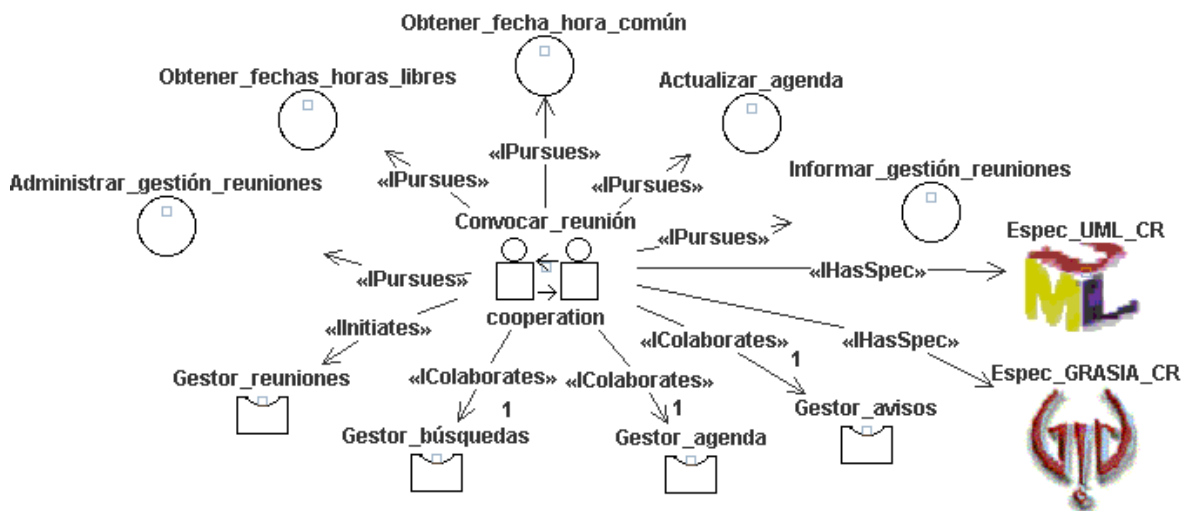


Figura 4.13 Interacción para detallar la realización de caso de uso Convocar Reunión

La interacción *Convocar\_reunión* se inicia por medio del rol *Gestor\_reuniones* cuando el usuario desea convocar una reunión. Su naturaleza es de cooperación ya que requiere la colaboración del resto de roles para alcanzar los cinco objetivos planteados (Administrar gestión reuniones, obtener fechas y horas libres, obtener fecha y hora común, actualizar agenda, informar gestión reuniones). Los roles y objetivos han sido identificados de los casos de uso detallados.

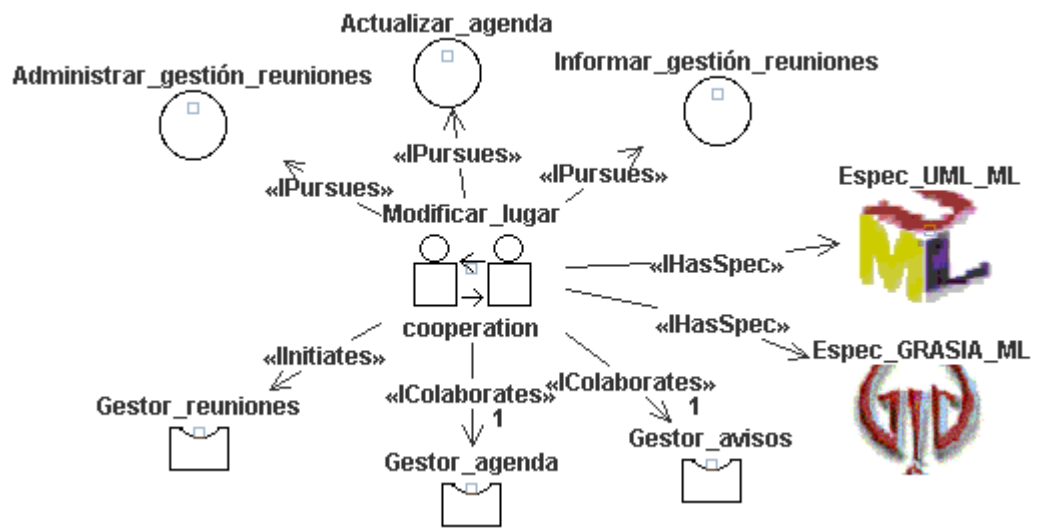


Figura 4.14 Interacción para detallar la realización de caso de uso Modificar lugar

La interacción *Modificar\_lugar* se inicia cuando un usuario que convocó una reunión desea modificar el lugar de la misma. Su naturaleza es de cooperación ya que cuando el rol *Gestor\_reuniones* inicia la interacción, solicita la colaboración del resto de roles para alcanzar los objetivos.

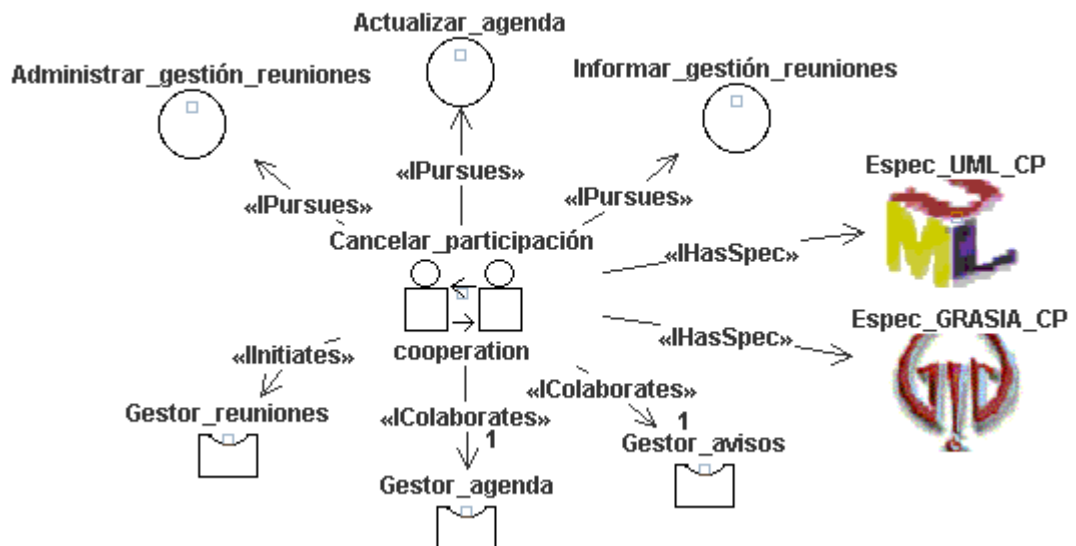


Figura 4.15 Interacción para detallar la realización de caso de uso Cancelar Participación

La interacción *Cancelar\_participación* se inicia por medio del rol *Gestor\_reuniones* cuando un usuario invitado a una reunión desea cancelar su participación. Su naturaleza es de cooperación, y existen tres objetivos a perseguir que son: administrar gestión reuniones, actualizar agenda e informar gestión reuniones.

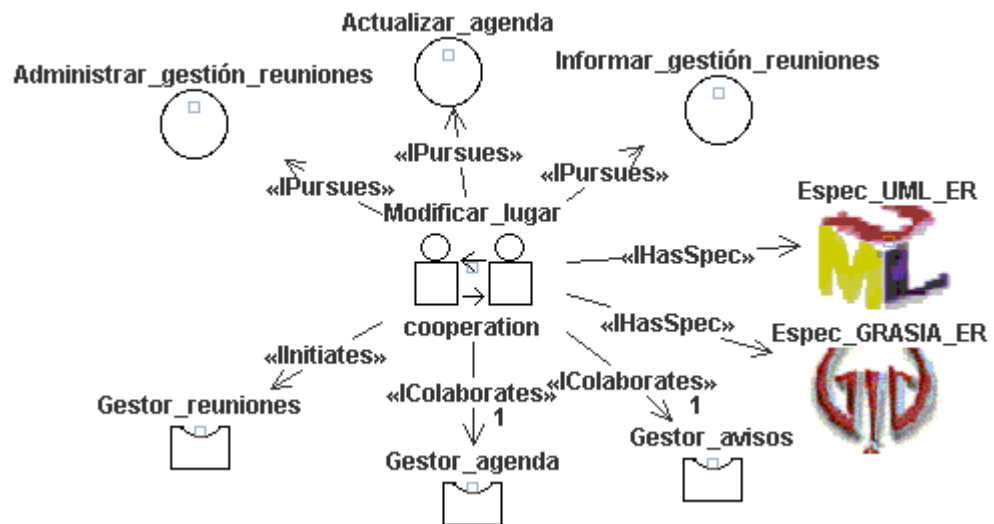


Figura 4.16. Interacción para detallar la realización de caso de uso Cancelar Reunión

La interacción *Cancelar\_reunión* se inicia cuando un usuario convocante de una reunión desea cancelar la misma. De igual manera el rol *Gestor\_reuniones* es el que recibe las peticiones del usuario e inicia la interacción. Su naturaleza es de cooperación ya que necesita de las funcionalidades del resto de roles para alcanzar los objetivos.

Para lograr los objetivos presentados en los modelos de interacción se plantean los diagramas de colaboración que utiliza las expresiones secuenciales de UML (Espec\_UML).





Figura 4.17 Diagrama de colaboración para la interacción Convocar Reunión

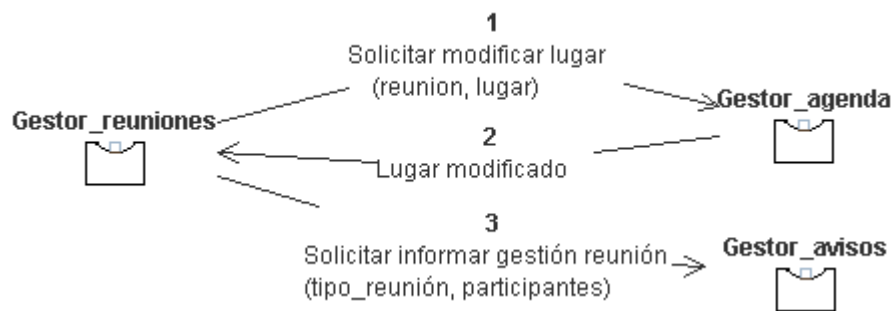


Figura 4.18 Diagrama de colaboración para la interacción Modificar lugar

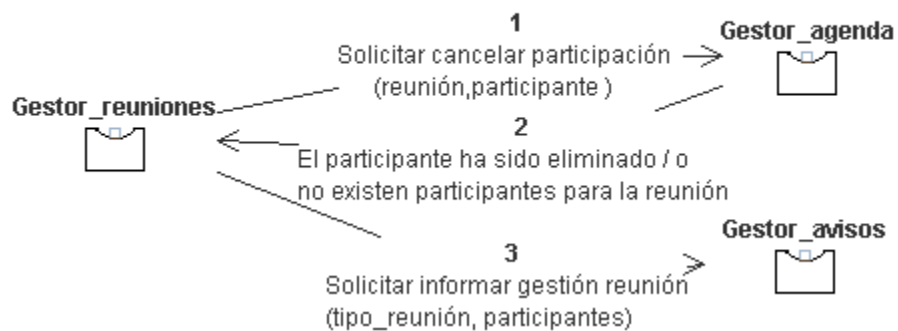


Figura 4.19 Diagrama de colaboración para la interacción Cancelar participación

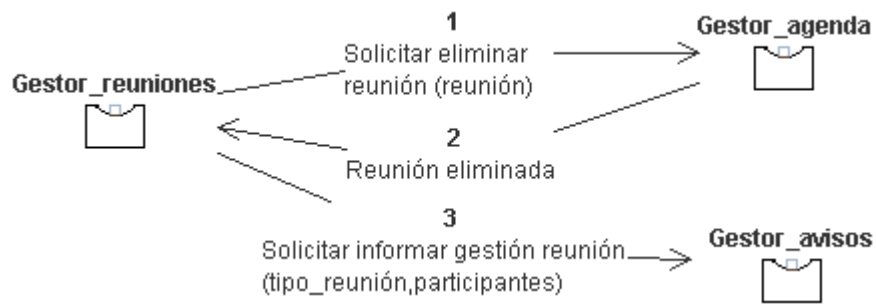


Figura 4.20 Diagrama de colaboración para la interacción Cancelar Reunión

#### 4.2.6.- META-MODELO DE OBJETIVOS Y TAREAS

Los objetivos encontrados en las interacciones y modelos de agente se estructuran utilizando un modelo de tareas y objetivos.

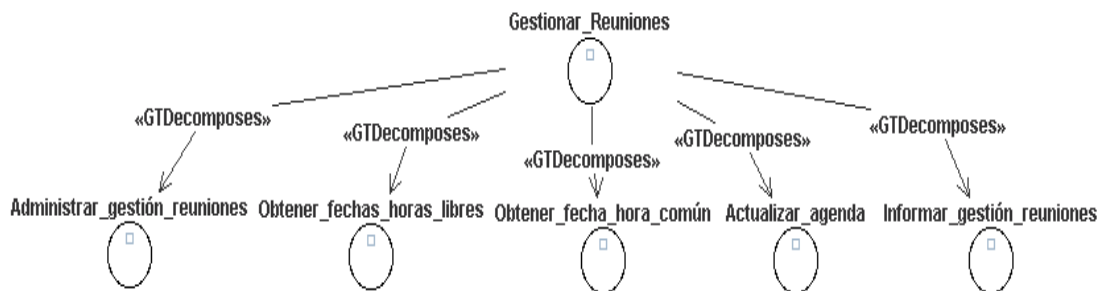
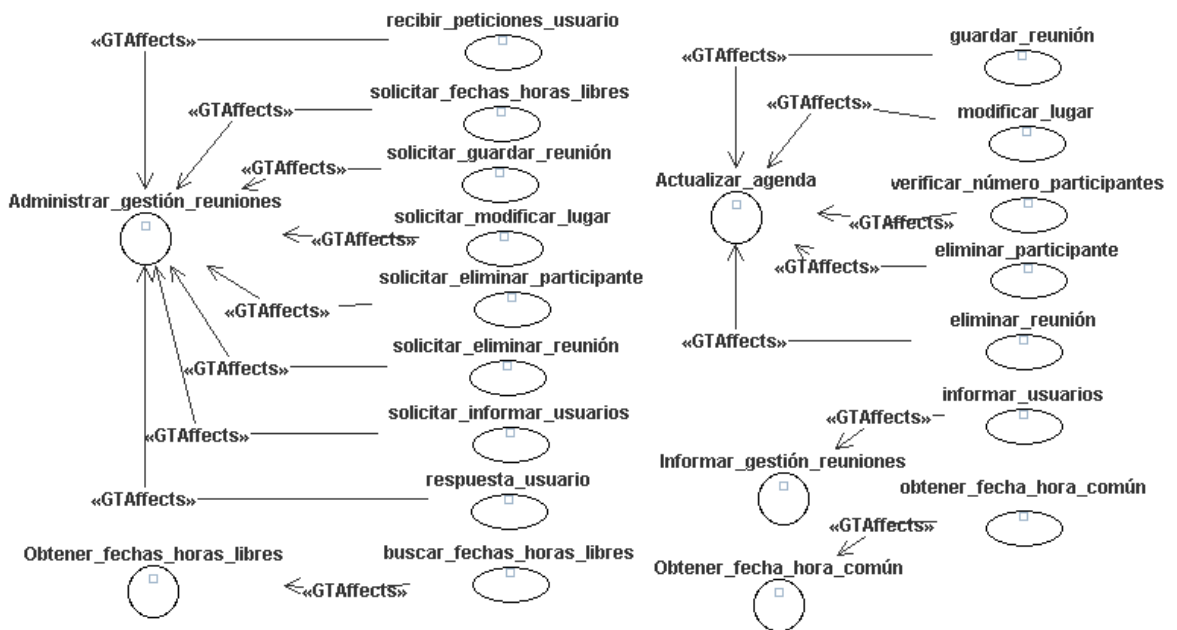


Figura 4.21 Estructuración inicial de objetivos

La gestión de reuniones (objetivo *Gestionar\_Reuniones*) se descompone en otros objetivos:

- *Administrar\_gestión\_reuniones*: Este objetivo permitirá, que cuando se realice cualquier operación de la gestión de reuniones (convocar, modificar o cancelar) el agente gestor de reuniones controle la interacción por medio de su rol, recibiendo las solicitudes del usuario y solicitado a cada agente su servicio, para de esta manera trabajar conjuntamente y así alcanzar el resto de objetivos.
- *Obtener\_fechas\_horas\_libres*: Cuando un usuario desea convocar una reunión se necesita conocer todas las fechas y horas libres de los invitados.

- *Obtener\_fecha\_hora\_común*: Una vez recibidas las fechas y horas libres de los invitados se procede a obtener la fecha y hora común para la reunión.
- *Actualizar\_agenda*: Permite guardar, modificar o eliminar el participante o la reunión de la base de datos cuando se realice cualquier operación de la gestión de reuniones.
- *Informar\_gestión\_reuniones*: Permite informar a los participantes de la reunión, que tipo de operación de la gestión de reuniones se realizó.

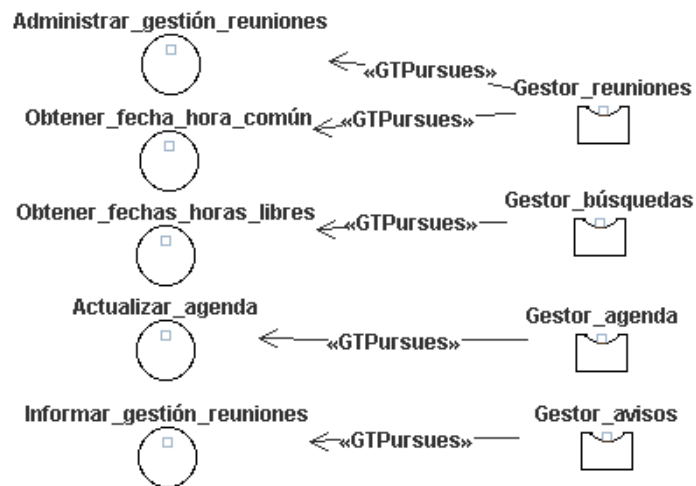


**Figura 4.22 Tareas asociadas a objetivos**

La identificación de tareas ha sido guiada por los objetivos que se han encontrado en los modelos de interacción. Para cada objetivo se han creado tareas cuya ejecución pueda enmarcarse dentro de los diagramas de colaboración anteriores.

- *recibir\_peticiones\_usuario*: Esta tarea se encarga de recibir las peticiones del usuario, ya sea convocar o eliminar reunión, modificar lugar o cancelar participación.
- *solicitar\_fechas\_horas\_libres*: Esta tarea consiste en solicitar al agente buscador las fechas y horas libres de los invitados, después de que se haya recibido la petición del usuario de convocar una reunión.

- *buscar\_fechas\_horas\_libres*: Recibido la solicitud, los datos de la reunión y de los invitados, se procede a buscar la fechas y horas libres de cada uno de ellos.
- *obtener\_fecha\_hora\_común*: Busca la fecha y hora común de entre las fechas y horas libres de los invitados que la ha facilitado el agente buscador.
- *solicitar\_guardar\_reunión*: La función de esta tarea es de solicitar al agente agenda que guarde la información de la nueva reunión.
- *guardar\_reunión*: Esta tarea almacena los datos de la reunión en la base de datos.
- *solicitar\_informar\_usuarios*: Tras haber, guardado, modificado o eliminado una reunión, se procede a solicitar al agente informador comunique a los usuarios.
- *informar\_usuarios*: Informa a los usuarios que han participado en las reuniones que tipo de gestión se ha realizado.
- *respuesta\_usuario*: Se encarga de mandar la respuesta al usuario de la acción escogida.
- *solicitar\_modificar\_lugar*: La función de esta tarea es de solicitar al agente agenda que modifique el lugar de la reunión.
- *modificar\_lugar*: Una vez recibido la petición de modificar el lugar y la nueva información se procede a realizar el cambio en la base de datos.
- *verificar\_número\_participantes*: Cuando un usuario desea cancelar su participación en una reunión existente, se debe comparar el número mínimo de invitados (dato ingresado por el usuario al convocar la reunión), con el número de participantes que constan actualmente en la reunión. Si existe aún el número mínimo de invitados procede a cancelar la participación del usuario, caso contrario procede a solicitar que la reunión se cancele.
- *solicitar\_eliminar\_invitado*: En seguida de que se haya realizado la tarea *verificar\_número\_participantes*, el agente agenda elimina el invitado de la base de datos de la reunión correspondiente.
- *eliminar\_invitado*: Tras haber recibido la solicitud de eliminar invitado y el dato del usuario que no puede asistir, se procede a modificar la base de datos.
- *solicitar\_eliminar\_reunión*: Esta tarea consiste en solicitar al agente agenda que elimine la reunión.
- *eliminar\_reunión*: Recibido la solicitud y el dato de la reunión, esta tarea se encarga de eliminarla de la base de datos.



**Figura 4.23 Asociaciones entre objetivos y roles**

La participación de los roles en las interacciones está sujeta a la relación entre los objetivos que persiguen y los asociados a la interacción.

A continuación se detallan los roles:

- *Gestor\_reuniones*: Este rol se hace responsable de todas las tareas que se realizan para alcanzar el objetivo de administrar la gestión de reuniones conjuntamente con el de obtener la fecha y hora común para la misma.
- *Gestor\_búsquedas*: La motivación de este rol es de obtener las fechas y horas libres, por lo tanto se hace responsable de las diferentes tareas que le corresponden para alcanzar este objetivo.
- *Gestor\_agenda*: Este rol se encarga de gestionar la funcionalidad del agente agenda, para alcanzar el objetivo designado.
- *Gestor\_avisos*: Se encargará de informar a los usuarios por medio de las tareas asignadas al objetivo a perseguir.

#### **4.2.7.- META-MODELO DE ENTORNO**

Una vez decidido cómo se alcanzan los objetivos, surgen nuevas necesidades que han de representarse con modelos de entorno. Se trata de las aplicaciones requeridas por las tareas para ejecutar su cometido.

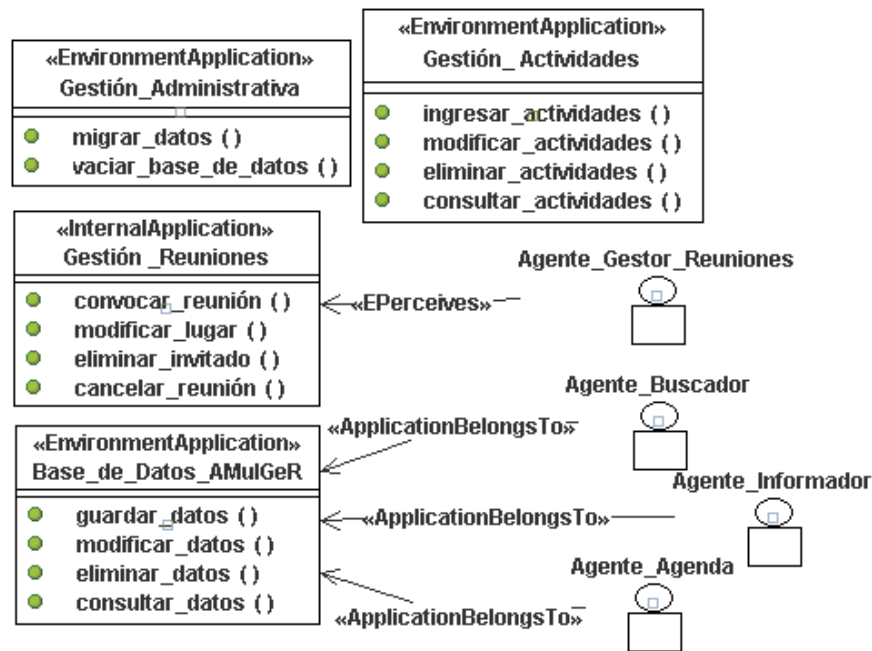


Figura 4.24 Elementos identificados en el entorno

El sistema AMulGeR esta formado por tres gestiones: la gestión administrativa, la gestión de actividades y la gestión de reuniones representadas en el modelo por las aplicaciones externas *Gestión\_Administrativa*, *Gestión\_Actividades* y la aplicación interna *Gestión\_Reuniones* respectivamente.

Cada aplicación tiene asociado operaciones que se han obtenido de los requisitos del sistema. Además, solamente en la gestión de reuniones aparece la intervención de los agentes, por lo que el *Agente\_Gestor\_Reuniones* percibirá las solicitudes de los usuarios cuando desean gestionar una reunión.

Se ha identificado una aplicación externa denominada *Base\_de\_Datos\_AMulGeR* (que representa a la base de datos del sistema) para poder dar a conocer que los agentes: buscador, agenda e informador la manipulan para alcanzar sus objetivos.

#### 4.2.8.- META-MODELO DE ORGANIZACIÓN

Para terminar el análisis, el modelo de organización recoge todos los resultados de los modelos anteriores y procede a su estructuración.

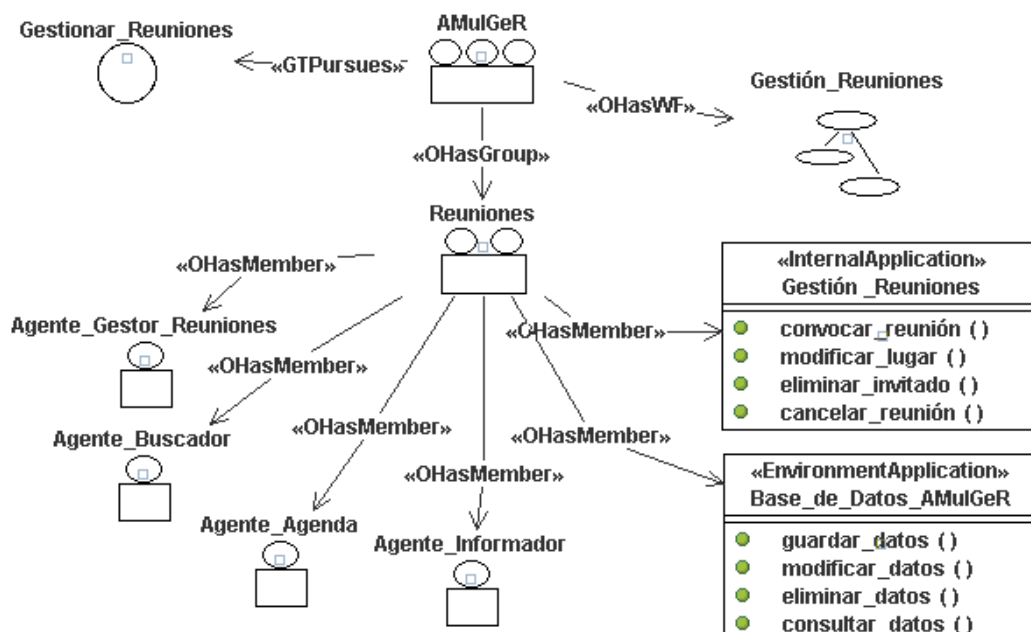


Figura 4.25 Representación del sistema Multiagente para el sistema Gestión de Reuniones (AMulGeR)

Dentro de la organización que se la ha denominado *AMulGeR* se distingue un grupo llamado *Reuniones* y un flujo de trabajo denominado *Gestión Reuniones*. El primero agrupa los agentes: gestor de reuniones, buscador, agenda e informador, la aplicación interna de gestión de reuniones y la aplicación externa de la base de datos, ya que todos estos tienen en común un solo objetivo que es gestionar las reuniones.

Toda la organización tiene un flujo de trabajo donde se detalla la descripción funcional del sistema. La existencia de este flujo se debe a la necesidad de organizar las tareas que se han identificado.

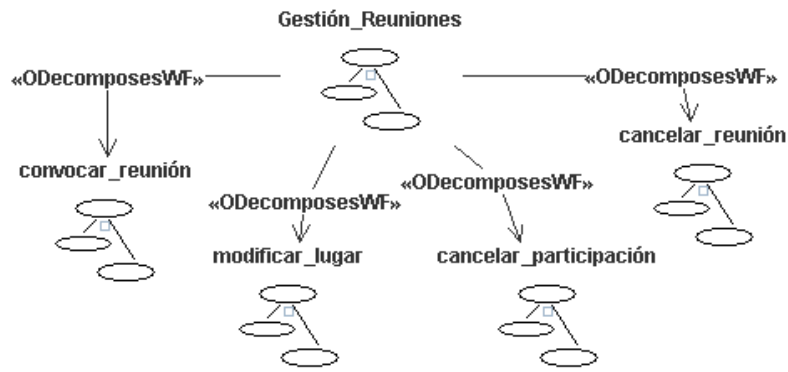


Figura 4.26 Refinamiento del flujo de trabajo Gestión de Reuniones

El flujo de trabajo *Gestión\_Reuniones* se refina distinguiendo entre *convocar\_reunión*, *modificar\_lugar*, *eliminar\_participante* e *informar\_usuarios*. En la etapa de diseño se ofrece más detalles respecto a la descomposición en tareas de éstos flujos de trabajo.

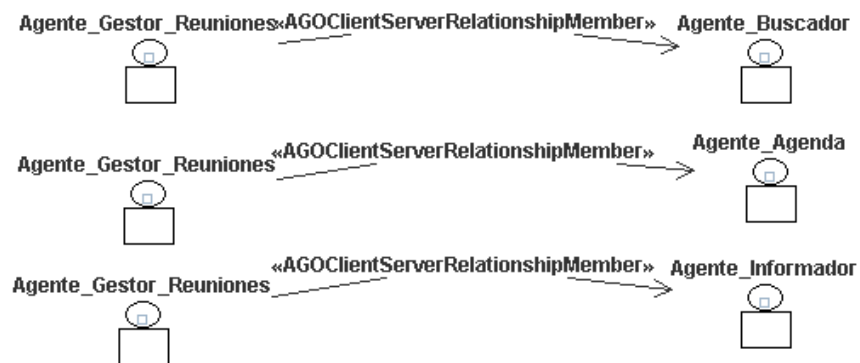


Figura 4.27 Relaciones sociales entre agentes

Las relaciones sociales entre los agentes es de Cliente-Servidor ya que el agente gestor reuniones solicita al resto de agentes el servicio que ofrecen para alcanzar los objetivos.

#### 4.2.9.- CONSTRUCCIÓN DE LA ONTOLOGÍA

En este sistema la ontología cubre toda la información sobre la gestión de reuniones como es para convocar y eliminar una reunión, modificar el lugar y cancelar participación, dentro del cual se ha identificado los siguientes conceptos:



**Reunión:** con este concepto se cubre toda la información necesaria para representar a una reunión. Los atributos necesarios son:

- *descripción:* es el detalle de la reunión
- *lugar:* lugar donde se va realizar la reunión
- *fecha:* fecha en la que se lleva a cabo la reunión
- *hora\_inicio:* hora en la que comienza la reunión.
- *hora\_fin:* hora en la que termina la reunión
- *num\_min\_participantes:* para poder celebrar una reunión es necesario un número mínimo de participantes, que es lo que indica este atributo.
- *id\_convocante:* es el identificador del usuario que convocó la reunión.
- *participantes:* lista de identificadores de los usuarios del sistema que van a intervenir en la reunión.

**FH\_Posible\_Reunion:** este concepto se utiliza para representar un intervalo de hora en una fecha posible. Sus atributos son:

- *fecha\_posible:* fecha en la que un usuario participante a una reunión tiene horas libres.
- *hora\_inicial\_posible:* hora inicial libre de un intervalo
- *hora\_final\_posible:* hora final libre de un intervalo.

**Horas\_participante:** se utiliza para representar las fechas y horas libres de un cierto participante a una reunión. Necesita los siguientes atributos:

- *FH\_Posibles:* lista de la información del concepto FH\_Posible\_Reunion
- *invitado:* identificador del usuario participante en la reunión

Además de conceptos en una ontología también es necesario representar las acciones que tienen que realizar los agentes:

**Convocar\_reunión:** se utiliza esta acción para convocar una reunión. Los atributos necesarios para realizar esta acción son:

- *descripción:* es el detalle de la reunión a convocar
- *lugar:* lugar donde se va a realizar la reunión

- *fecha\_min* y *fecha\_max*: fechas en las cuales se desea que se convoque la reunión.
- *duración*: es el tiempo que durará la reunión
- *num\_min\_participantes*: para poder celebrar una reunión es necesario un número mínimo de participantes, que es lo que indica este atributo.
- *id\_convocante*: es el identificador del usuario que convoca la reunión.
- *participantes*: lista de identificadores de los usuarios del sistema que van a intervenir en la reunión.
- *prioridad*: este atributo indica si la prioridad de la reunión a convocar es alta o baja. En la prioridad alta, no se procede a buscar fechas y horas libres, se realiza la reunión en la fecha y hora designada por el usuario convocante. En cambio en la prioridad baja se procede a buscar las fechas y horas libres de los participantes, respetando sus horarios, entre las fechas escogidas por el convocante.
- *fecha\_reunión*: fecha en la que se va a realizar la reunión
- *hora\_reunión*: hora en la que se va a realizar la reunión

***Buscar\_fh\_libres***: se utiliza para buscar las fechas y horas libres de una lista de usuarios participantes de una reunión a convocarse. Sus atributos son:

- *prioridad*: factor (alta o baja) que influye en la búsqueda.
- *participantes*: lista de identificadores de los usuarios del sistema de quienes se va a proceder a buscar las fechas y horas libres.
- *fecha\_max* y *fecha\_min*: fechas, entre las cuales se procederá a buscar las fechas y horas libres.

***Modificar\_lugar***: se utiliza para modificar el lugar de una reunión. Tiene como atributos:

- *codigo\_reunion*: Identificador de la reunión que se quiere modificar.
- *lugar*: dato del nuevo lugar.

***Eliminar\_participante***: se utiliza esta acción cuando un usuario invitado a una reunión desea cancelar su participación. Sus atributos son:

- *codigo\_reunion*: Identificador de la reunión de la cual se va a eliminar el participante.
- *invitado*: identificador del participante que cancela su participación en una reunión.

***Eliminar\_reunion***: esta acción se utiliza cuando un usuario convocante de una reunión desea cancelar la misma. Posee un solo atributo.

- *codigo\_reunion*: Identificador de la reunión a eliminar.

Se tiene además predicados en esta ontología:

***Respuesta\_FHLibres***: este predicado se utiliza para informar las fechas y horas libres de todos los participantes. Su atributo es:

- *informar*: lista del concepto Horas\_participante.

***Informar\_participantes***: se utiliza para informar a los usuarios sobre la gestión de reuniones. Sus atributos son:

- *descripción*: detalle de la gestión de reuniones
- *codigo\_reunion*: Identificador de la reunión que se esta gestionando
- *motivo*: El tipo de la gestión de reuniones que se esta realizando
- *leido*: para conocer si el usuario ha leído el mensaje de información
- *participantes*: lista lista de identificadores de los usuarios a los que se debe informar sobre la gestión de reuniones

## 4.3.- DISEÑO

### 4.3.1.- META-MODELO DE AGENTE

Los modelos de agente no se modifican en este caso

### 4.3.2.- META-MODELO DE INTERACCIONES

Los diagramas de colaboración de la etapa anterior se representan con un conjunto de diagramas GRASIA. La conversión inicia traduciendo pasos de mensajes a unidades de interacción.

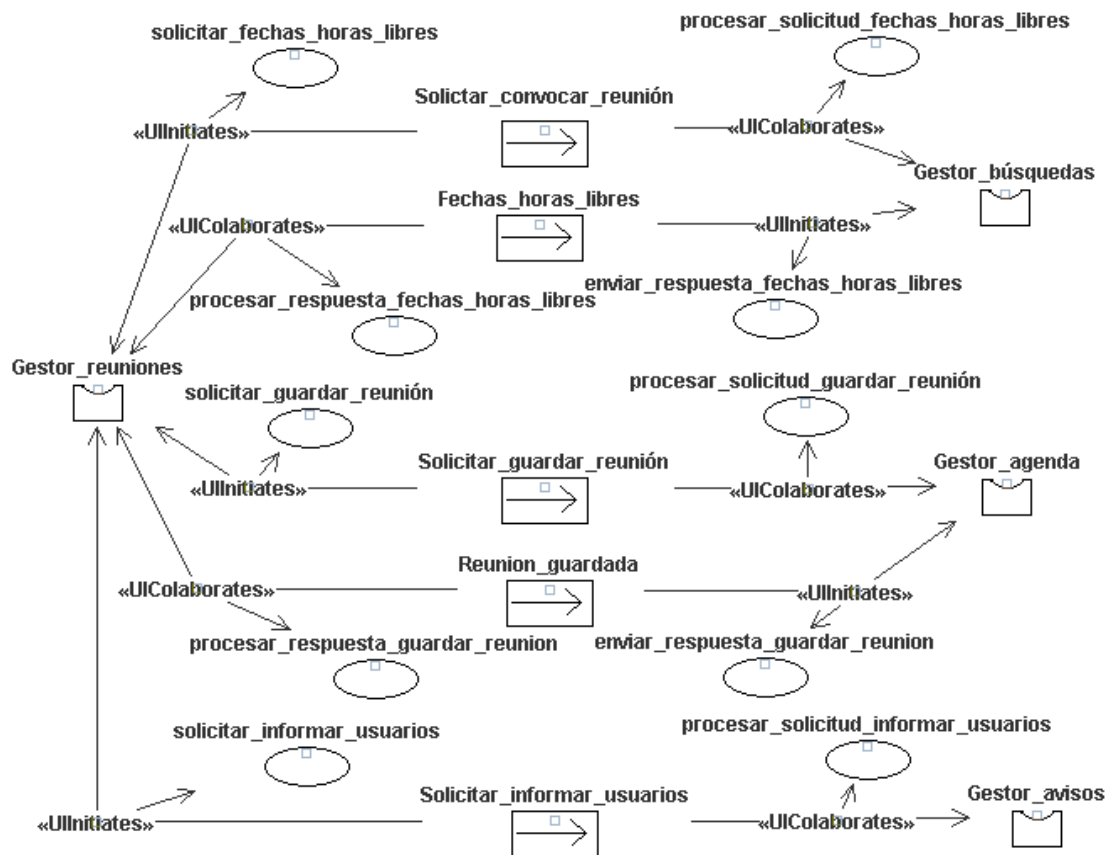


Figura 4.28 Unidades de interacción identificadas para la Interacción Convocar Reunión





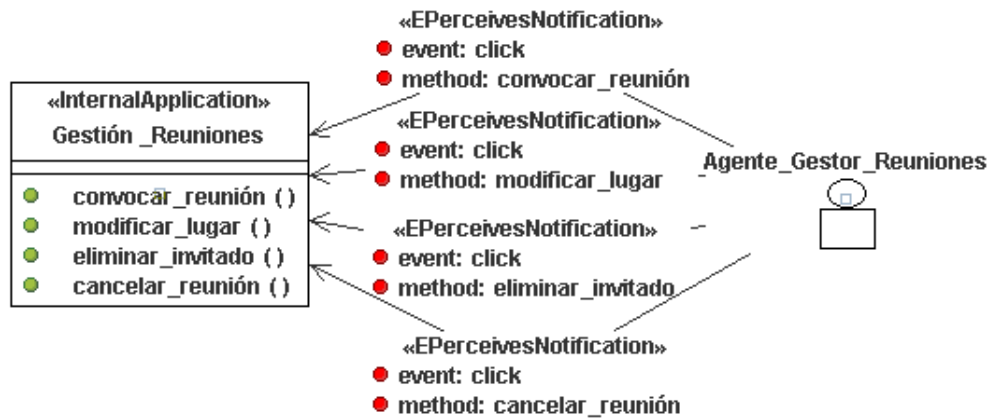


Figura 4.33 Percepción del Agente Gestor de Reuniones

#### 4.3.5.- META-MODELO DE ORGANIZACIÓN

Los modelos de organizaciones se refinan asociando tareas a los flujos de trabajo.

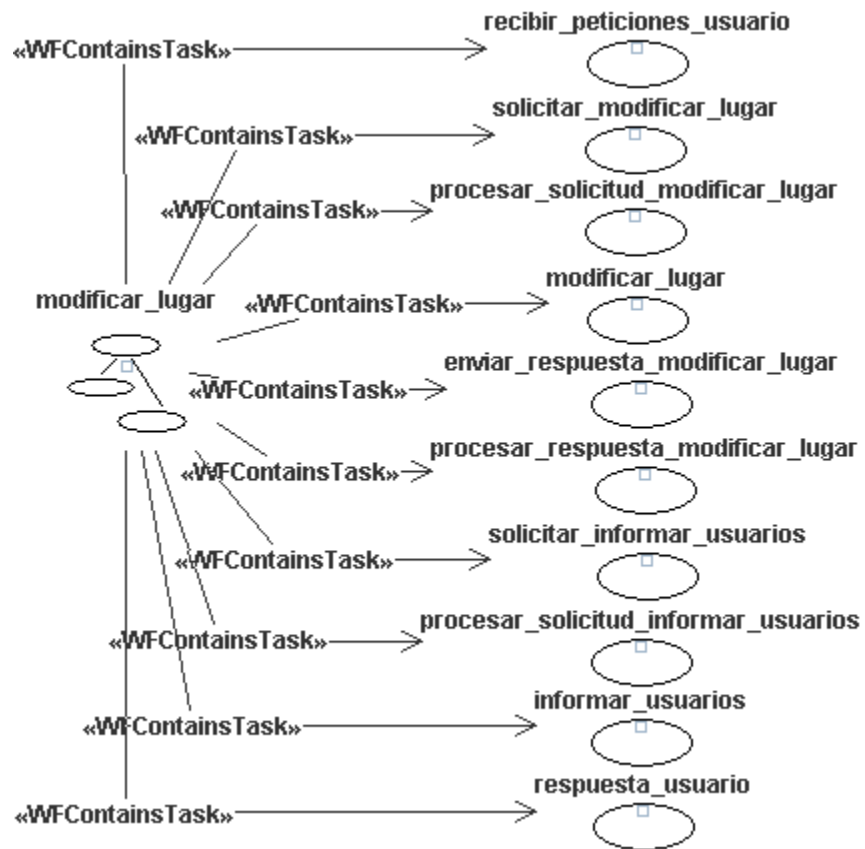


Figura 4.34 Tareas que componen el flujo de trabajo Modificar Lugar

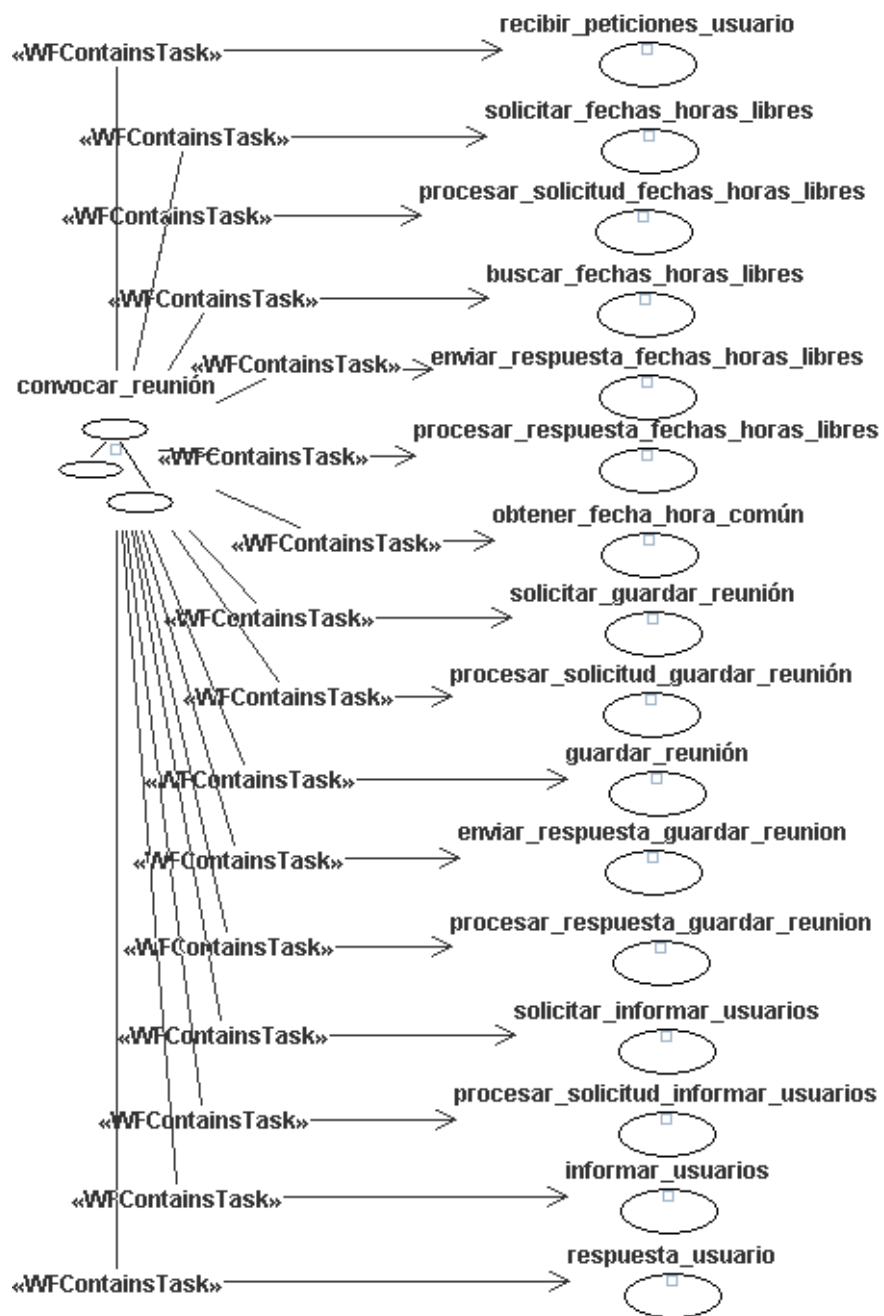


Figura 4.35 Tareas que componen el flujo de trabajo Convocar Reunión



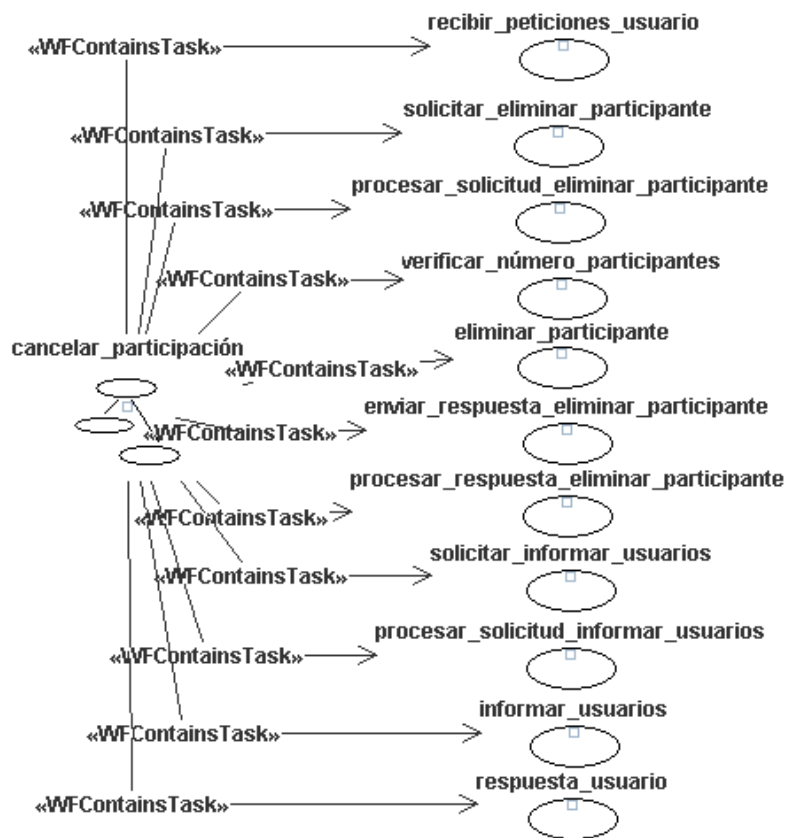


Figura 4.36 Tareas que componen el flujo de trabajo Cancelar Participación

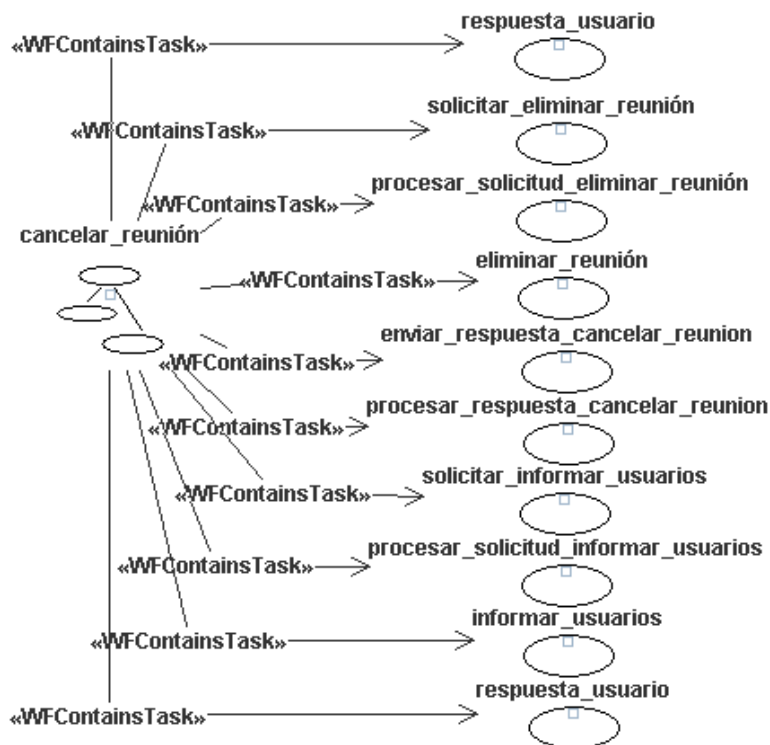


Figura 4.37 Tareas que componen el flujo de trabajo Cancelar Reunión

A continuación se presentan las dependencias entre las tareas para cada flujo de trabajo. Una tarea no se puede realizar si la que le antecede no ha sido ejecutado.

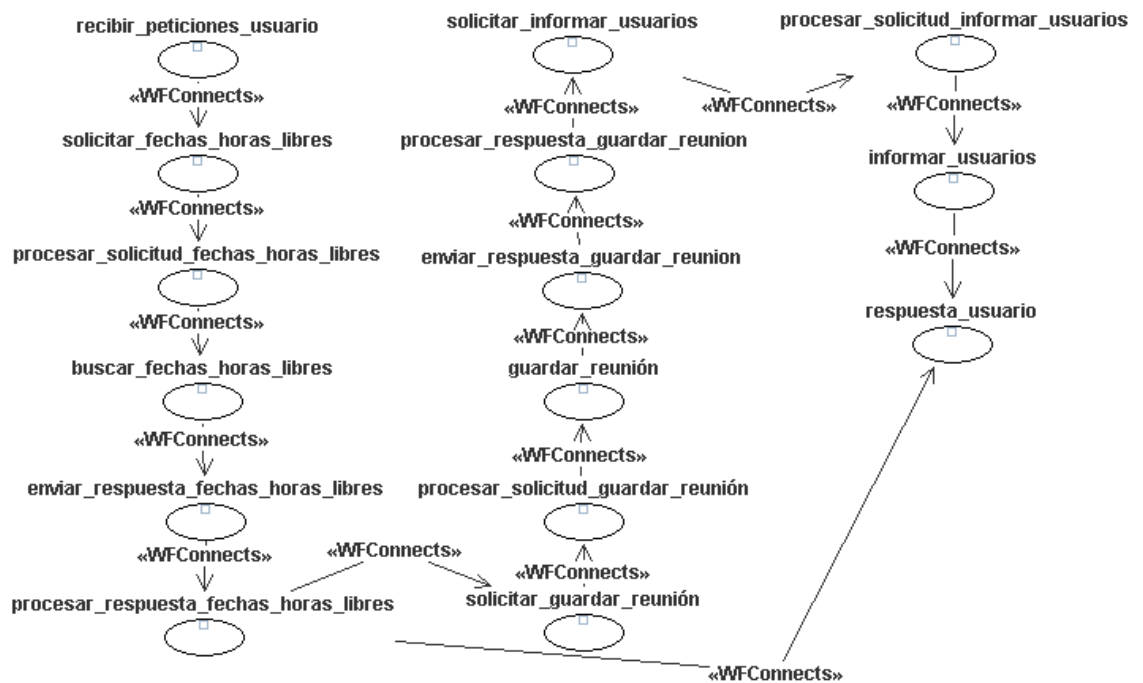


Figura 4.38 Dependencias entre las tareas del flujo Convocar Reunión

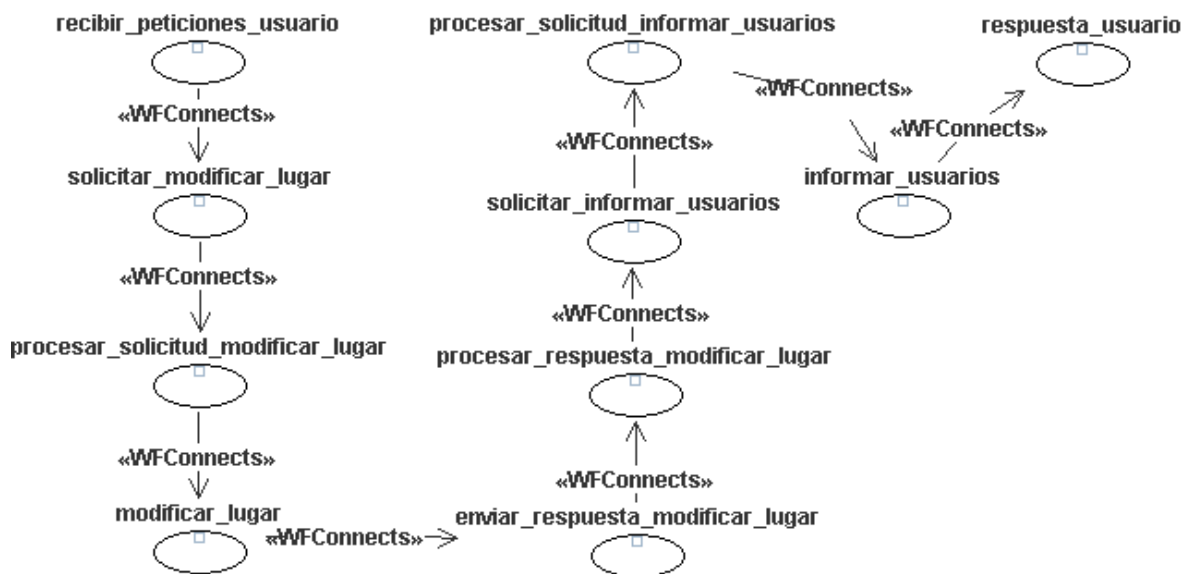


Figura 4.39 Dependencias entre las tareas del flujo Modificar Lugar

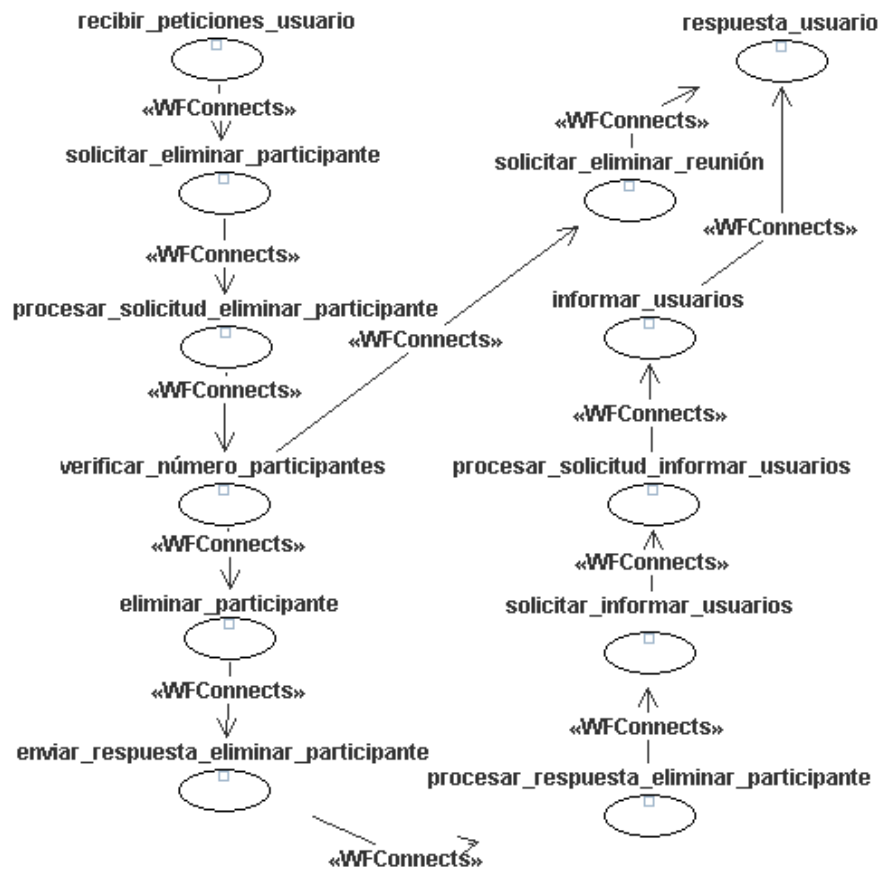


Figura 4.40 Dependencias entre las tareas del flujo Cancelar Participación

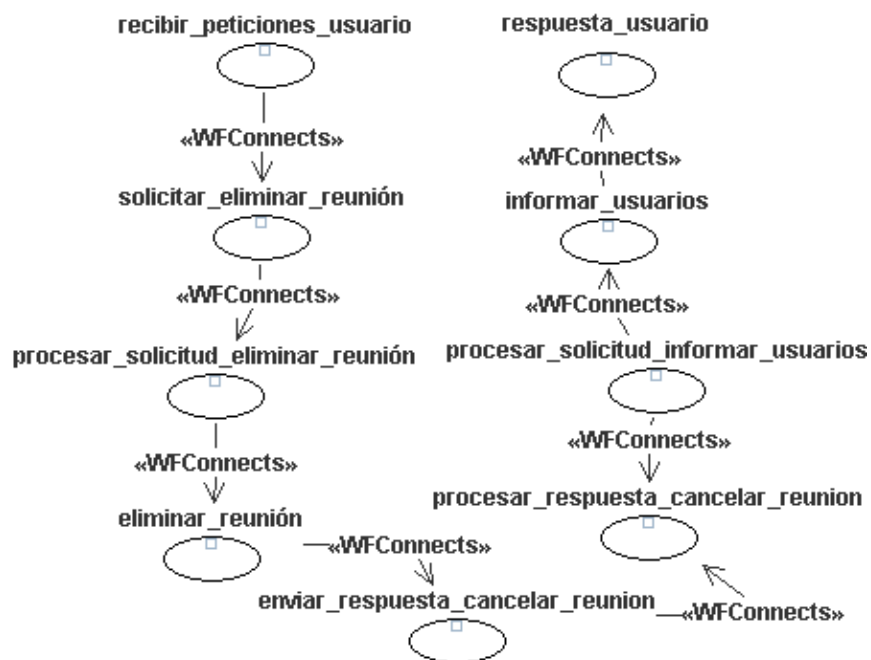


Figura 4.41 Dependencias entre las tareas del flujo Cancelar Reunión

La asociación de los roles con las tareas permite identificar quién es responsable de ejecutarlas

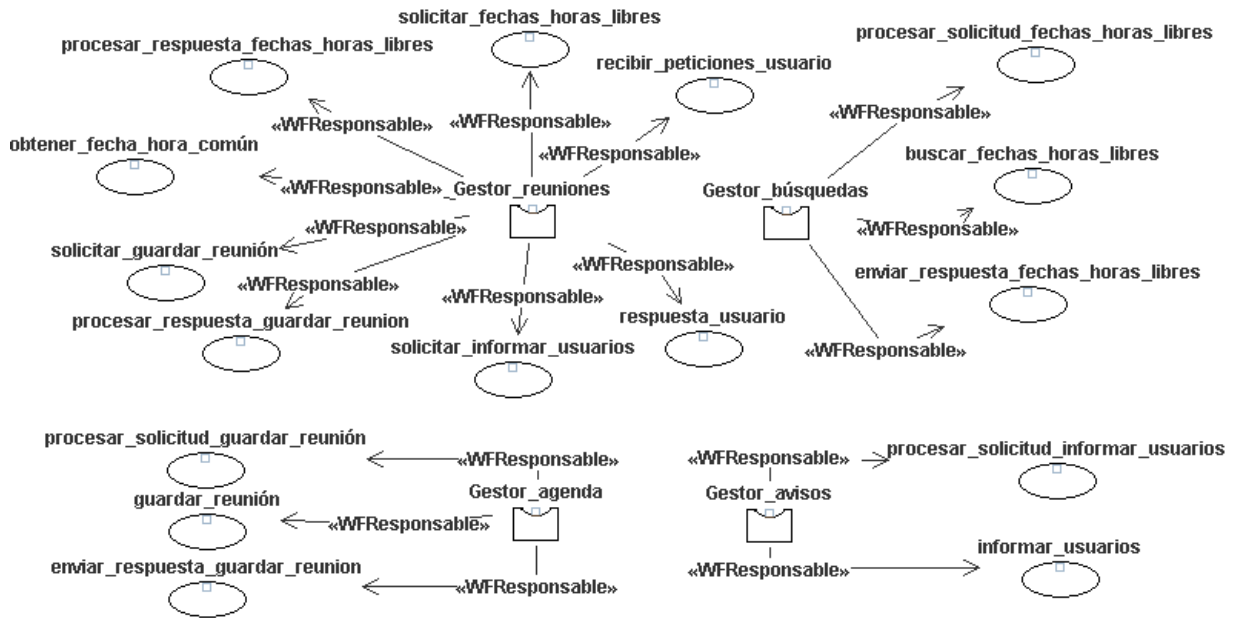


Figura 4.42 Responsables de la ejecución de tareas en el flujo de trabajo Convocar Reunión

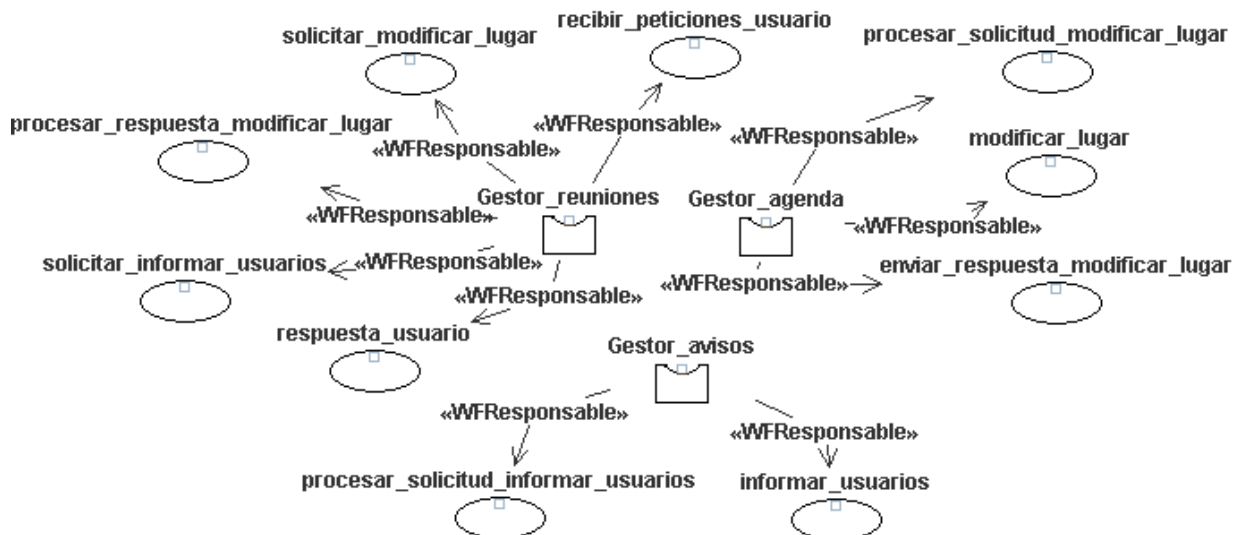


Figura 4.43 Responsables de la ejecución de tareas en el flujo de trabajo Modificar Lugar

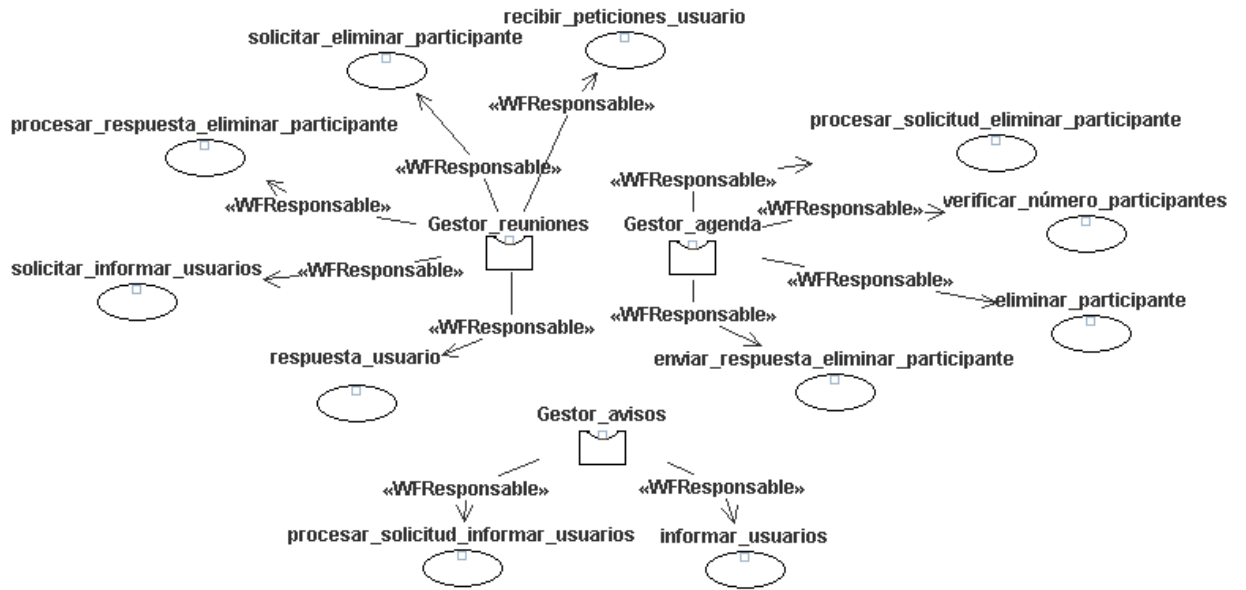


Figura 4.44 Responsables de la ejecución de tareas en el flujo de trabajo Cancelar Participación

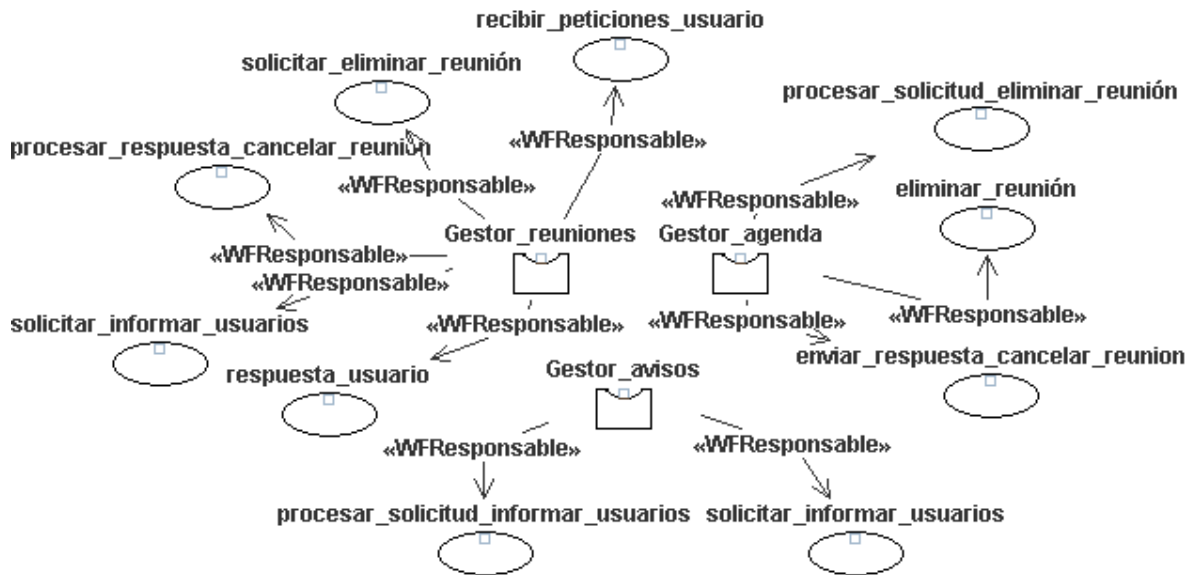


Figura 4.45 Responsables de la ejecución de tareas en el flujo de trabajo Cancelar Reunión

De los casos de uso se identifican las tareas que producen las interacciones principales.

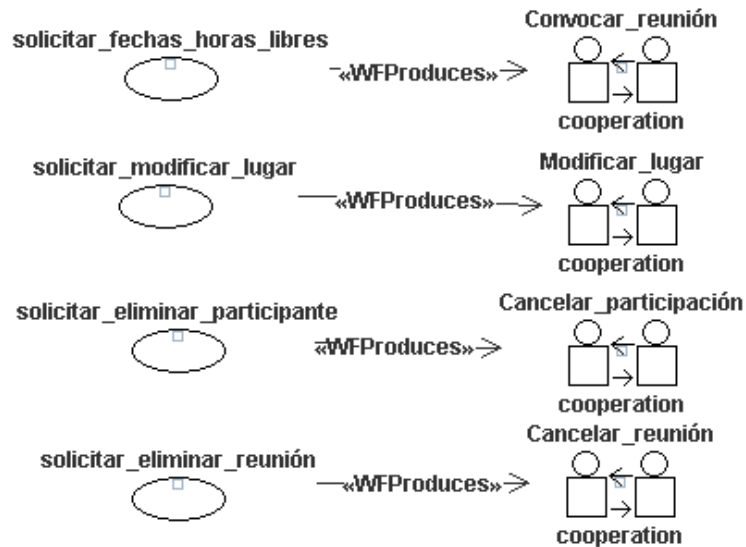
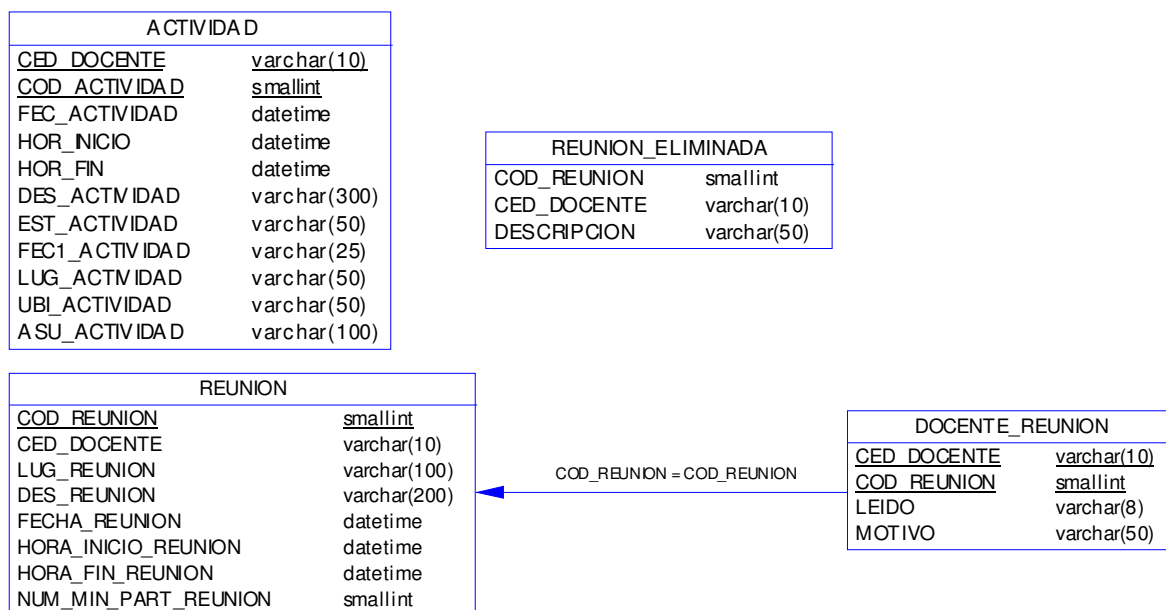


Figura 4.46 Tareas que producen interacciones

#### 4.3.6.- MODELO ENTIDAD RELACION

En este modelo se encuentran los campos de las tablas de la base de datos migrada del sistema escolástico de la ESPE-L al sistema AMulGeR. La información de estas tablas no puede ser publicada debido a que es información reservada, si desea conocerla puede acercarse a la Unidad de Tecnologías de Información y Comunicación (TIC) de la ESPE-L.

A la base de datos mencionada anteriormente se agregó cuatro tablas que se relacionan entre sí y con algunas de las migradas. La Figura 4.47 muestra el modelo físico de aquellas tablas con sus respectivas relaciones.



**Figura 4.47 Modelo Entidad Relación**

## 4.4.- IMPLEMENTACIÓN

Una vez terminado el análisis y el diseño del sistema AMulGeR (Agenda Multiagente para la Gestión de Reuniones) orientado a la Web, se procede a su desarrollo. A continuación se detallan las tecnologías utilizadas en las tres gestiones que conforman el sistema.

### 4.4.1.- SELECCIÓN DE LA BASE DE DATOS

La base de datos utilizada fue MySQL (My Structured Query Language) Server 5.0. Uno de los factores importantes que influyó en esta selección es que el sistema AMulGeR debe adecuarse a los sistemas informáticos utilizados en la ESPE-L (que son actualmente administrados en la Unidad de Tecnologías de Información y Comunicación, *TIC*), además porque es uno de los Sistemas Gestores de Bases de Datos más populares desarrolladas bajo la filosofía de código abierto, puede usarse gratuitamente, es flexible, seguro, fácil de instalar, utilizar, configurar y provee una solución robusta a los usuarios que la manipulan.

#### **4.4.2.- GESTIÓN ADMINISTRATIVA**

Esta gestión realiza la migración de los datos correspondientes a: docente, facultad, período académico, materia, aula, paralelo, día, horarios que actualmente están siendo administrados por medio de una base de datos denominada Sybase a la base de datos utilizada (MySQL) por el sistema AMulGeR.

Para la codificación, se usó la plataforma de desarrollo Microsoft Visual Studio .NET 2005, Lenguaje C#, ya que es una nueva y potente tecnología de alto nivel de productividad para escribir páginas web dinámicas. Para establecer la conexión entre la base de datos MySQL y esta plataforma de desarrollo se hizo uso del conector: MySQL Connector Net 5.0.7 que permite a los desarrolladores crear fácilmente aplicaciones .NET que requieren seguridad y alto rendimiento de datos.

Para el buen funcionamiento de esta gestión se integró las tecnologías mencionadas anteriormente con el Servidor Web: Internet Information Sever (*IIS*) porque ofrece un elevado nivel de funcionalidad, es seguro, confiable y además es un producto desarrollado por Microsoft lo que facilita el trabajo en conjunto con plataformas de desarrollo de su misma organización.

La gestión administrativa fue desarrollada con tecnología utilizada también en los sistemas informáticos de la ESPE-L.

#### **4.4.3.- GESTIÓN DE ACTIVIDADES**

El diseño de la interfaz, funcionalidades y la implementación de esta gestión se lo ha realizado en el lenguaje de programación PHP (*PHP: Hypertext Preprocessor*) 5.0 integrado con el lenguaje de programación JavaScript siguiendo los estándares del sistema portal académico de la ESPE-L. PHP es un lenguaje usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios web y JavaScript es un lenguaje que se ejecuta en la máquina del cliente y es muy útil para realizar validaciones de la información ingresada por el usuario.



Para visualizar las páginas web PHP se hace uso del Servidor Web Apache 2.2 que es el más difundido en Internet por su buen comportamiento y funcionamiento

El ambiente de trabajo usado fue la herramienta Dreamweaver 8, el cual es un editor de lenguajes de programación orientados a la web.

#### **4.4.4.- GESTIÓN DE REUNIONES**

Toda la codificación para esta gestión se la realizó en el lenguaje de programación JAVA JDK (Java Development Kit) 5.0.15, que incluye el API (*Application Programming Interface*) que es un conjunto de clases para programar en el lenguaje y el JRE (*Java Runtime Environment*) que es el compilador.

Para la codificación de los agentes software se utilizó el API de JADE 3.6 que está basado en el lenguaje de programación JAVA.

El JDK de Java no ofrece un ambiente de trabajo para proyectos complejos, por eso, se hizo uso del IDE (*Integrated Development Environment*) ECLIPSE 3.2 que es una plataforma de software de código abierto para desarrollar aplicaciones. Esta basado en complementos o plugins (aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica) y es independiente de una plataforma o sistema operativo para su funcionamiento. Para ejecutarla se necesita al menos la versión 3.4 del JRE o el JDK.

El servidor Web utilizado es Apache Tomcat 5.0 que es un contenedor de Servlets (componentes que se ejecutan en el servidor) que soporta tecnologías como JS (*Java Servlets*) y JSP (*Java Servlets Page*).

Para el correcto funcionamiento de la Agenda Multiagente es necesaria la conexión de la gestión de actividades y de reuniones. Esto se lo realizó mediante un Servlet que recibe las peticiones del usuario a través de la interfaz diseñada en PHP. A continuación el Servlet convoca a un componente de Jade llamado JadeGateway el cual crea

automáticamente un agente (AgenteGateway) que sirve de pasarela entre el Servlet y JADE, y además se encarga de enviar un mensaje ACL al Agente Gestor Reuniones y procesar su contestación enviándola de vuelta al Servlet. Este nuevo agente creado recibe la petición del usuario desde el Servlet por medio de un objeto.

Para mas detalle de la integración de JADE con un Servlet mediante la clase JadeGateway ver Anexo A.

#### **4.4.5.- SISTEMA OPERATIVO**

Todo el sistema AMulGeR se lo realizó bajo la plataforma Windows XP Service Pack 2.

#### **4.4.6.- FUNCIONAMIENTO DEL SISTEMA AMulGeR**

Como se pudo observar anteriormente las gestiones que componen el sistema AMulGeR están desarrolladas con tecnologías diferentes. A continuación se detalla el funcionamiento del sistema integradas las tres gestiones: administrativa, actividades y reuniones.

La parte administrativa debe ser manipulada por el administrador del sistema, y consiste en migrar los datos necesarios a la base de datos que usa el sistema AMulGeR y que se va a utilizar en la gestión de actividades y reuniones. Esta tarea debe realizarse al inicio de cada semestre o cuando la nueva información de docentes, horarios, materias, etc., estén ingresadas en la base de datos a migrar. Al finalizar el período académico el administrador deberá borrar la información generada durante todo el semestre de la base de datos, para repetir el procedimiento de migración.

Para el correcto funcionamiento del Sistema AMulGeR, se añadió información a la base de datos, en la tabla docentes, un campo denominado *rango\_docente* de tipo entero (int), que es utilizado en la gestión de reuniones para poder controlar las prioridades entre

los docentes cuando realicen convocatorias. Los rangos ingresados pertenecientes al personal de la Institución, son los siguientes:

- Rector: 1
- Subdirector: 2
- Subdirector de Docencia: 3
- Subdirector de Investigación y Desarrollo: 3
- Director de Departamento: 4
- Director de Carrera: 5

El resto del personal posee un rango inferior y no le es permitido convocar reuniones.

Además se añadió a la tabla `facultad_instituto` en los campos respectivos los siguientes datos: Todos, Directores, Coordinadores. Esto permite filtrar los nombres de los docentes por la información mencionada anteriormente y por la facultad a la que pertenecen cuando desean encontrarlos y seleccionarlos al convocar una reunión.

La gestión administrativa es independiente de la gestión de actividades y reuniones, es decir que éstas dos últimas no tienen ninguna conexión con la primera, ya que es uso exclusivo para la administración de la base de datos que el sistema AMulGeR utiliza.

La gestión de actividades y reuniones deben ser manejadas por el usuario. La primera se encarga de administrar los horarios de clases y actividades o tareas por día, de los docentes de la ESPE-L y la segunda de realizar reuniones entre ellos estableciendo una fecha y hora de acuerdo a las horas disponibles controladas en la gestión anterior. Cabe recordar que solamente en la gestión de reuniones se tiene la intervención de los agentes software.

Las interfaces de estas dos gestiones están realizadas en PHP, las páginas desarrolladas se alojan, por lo general, en la carpeta `htdocs` del directorio donde se haya instalado el servidor web Apache. En cambio la parte funcional de la gestión de reuniones que son las clases java se encuentran albergadas en la carpeta `wepapps` del directorio donde

se haya instalado el servidor web Apache Tomcat. Los puertos utilizados en los servidores web para recibir las peticiones del cliente son el 80 y 8080 respectivamente.

El proceso de conexión de estas dos gestiones es el siguiente: cuando un usuario selecciona cualquiera de las opciones convocar reunión, modificar lugar, cancelar participación o cancelar reunión, internamente y transparente para el usuario, el sistema invoca por medio de la acción (`action="http://localhost:8080/AMulGeR/ServletGateWay"`) de un formulario de la página web PHP a un servlet (ServletGateWay, codificado en JAVA) y éste a su vez se comunica con la plataforma JADE y los agentes software (ver Anexo A) para proceder a gestionar la reuniones. Para presentar el resultado al usuario se lo hace mediante código HTML desde una clase JAVA para posteriormente realizar un hipervínculo (`href = 'http://localhost:80/servicios_web/agenda/reuniones.php'`) a la página web PHP de la gestión de reuniones alojada en el servidor web Apache.

En la gestión de reuniones intervienen la participación de cinco agentes software: *AgenteReunión* (encargado de recibir la peticiones del usuario, administrar la gestión de reuniones solicitando los servicios que proporcionan el resto de agentes y obtiene la fecha y hora común para la reunión a realizarse), *AgenteBuscador* (ofrece el servicio de proporcionar la fechas y horas libres por día de los participantes a una reunión), *AgenteAgenda* (ofrece el servicio de actualizar la base de datos, es decir la agenda de los participantes, cuando se gestione reuniones), *AgenteInformador* (ofrece el servicio de informar a los participantes de una reunión, la gestión realizada, por medio de mensajes dentro de su agenda) y *MiAgenteGateWay* (que sirve de pasarela entre la interfaz de usuario y la plataforma JADE). Para más detalle del funcionamiento y la comunicación entre éstos agentes revisar el análisis y diseño del sistema AMulGeR.

#### **4.5.- PRUEBAS**

A continuación se presentan las pruebas de los casos que pueden darse en la Gestión de Reuniones al ejecutar el sistema AMulGeR. (**A**genda **M**ultiagente para la **G**estión de **R**euniones).

#### 4.5.1.- CASO 1

**Gestión reuniones:** Convocar reunión

**Descripción:** Reunión

**Lugar:** Departamento de Electrónica

**Prioridad:** Baja

**Fecha mínima:** 2008-07-21

**Fecha máxima:** 2008-07-22

**Tiempo de duración:** 01:30 horas

**Número mínimo de participantes:** 2

**Convocante:** Ingeniero Edison Espinosa

**Invitados:** Ingenieros: Raúl Cajas, Raúl Rosero, Alexandra Corral

#### ING. EDISON ESPINOSA

21-7-2008		22-7-2008	
HORARIO	LUNES	HORARIO	MARTES
07:00 - 07:15		07:00 - 07:15	
07:15 - 09:15	Computación	07:15 - 09:15	Computación
09:15 - 15:00		09:15 - 14:00	
15:00 - 17:00	Metodología de Desarrollo Sistemas	14:00 - 14:45	Redactar documentos
17:00 - 17:30		14:45 - 16:15	<b>REUNION</b>
17:30 - 18:30	Revisar documentos	16:15 - 18:00	
18:30 - 19:30		18:00 - 20:00	Defensa de tesis
19:30 - 21:30	Gestión de Procesos y Calidad	20:00 - 21:30	

Tabla 4.4 Horarios del Ing. Edison Espinosa en los días 21-07-2008 y 22-07-2008

#### ING. RAÚL CAJAS

21-7-2008		22-7-2008	
HORARIO	LUNES	HORARIO	MARTES
07:00 - 09:30		07:00 - 09:30	
09:30 - 11:30	Computación I	09:30 - 12:00	Computación I
11:30 - 12:00		12:00 - 14:00	Programación
12:00 - 14:00	Programación	14:00 - 14:45	<b>REUNION</b>
14:00 - 14:15		14:45 - 16:30	
14:15 - 15:00	Organizar curso de Visual NET	16:30 - 17:00	Revisar capítulos tesis
15:00 - 16:00		17:00 - 17:15	
16:00 - 17:00	Corregir pruebas	17:15 - 19:15	Sistemas Operativos
17:00 - 17:15		19:15 - 19:30	
17:00 - 19:15	Sistemas Operativos	19:30 - 21:30	Operación de Sistemas Multiusuarios
19:15 - 19:30			
19:30 - 21:30	Aplicación de BD		

Tabla 4.5 Horarios del Ing. Raúl Cajas en los días 21-07-2008 y 22-07-2008

**ING. RAÚL ROSERO**

21-7-2008		22-7-2008	
HORARIO	LUNES	HORARIO	MARTES
07:00 - 12:00		07:00 - 14:45	
12:00 - 14:00	Computación	14:45 - 16:30	<b>REUNION</b>
14:00 - 15:00			Aplicaciones Distribuidas II
15:00 - 17:00	Aplicaciones Distribuidas	17:15 - 19:15	
17:00 - 17:15		19:15 - 19:20	
17:15 - 19:15	Computación	19:20 - 21:00	Revisar trabajos
19:15 - 19:30		21:00 - 21:30	
19:30 - 20:30	Revisar trabajos		
20:30 - 21:30			

**Tabla 4.6 Horarios del Ing. Raúl Rosero en los días 21-07-2008 y 22-07-2008**

**ING. ALEXANDRA CORRAL**

21-7-2008		22-7-2008	
HORARIO	LUNES	HORARIO	MARTES
07:00 - 09:30		07:00 - 09:30	
09:30 - 11:00	Programación	09:30 - 12:00	Computación II
11:30 - 12:00		12:00 - 14:00	Programación
12:00 - 14:00	Computación II	14:00 - 15:00	
14:00 - 15:00			Arquitectura de Computadores II
15:00 - 17:00	Arquitectura de Computadores II	15:00 - 17:00	
17:00 - 17:30		17:00 - 18:30	
17:30 - 18:30	Revisar capítulos tesis	18:30 - 19:30	Corregir pruebas
18:30 - 21:30		19:30 - 21:30	

**Tabla 4.7 Horarios de la Ing. Alexandra Corral en los días 21-07-2008 y 22-07-2008**

**Respuesta:** Se ha convocado una nueva reunión el 2008-07-22 a las 14:45. El Ingeniero (a), Ingenieros (as): Corral Díaz María Alexandra Corral no puede asistir porque no dispone de una hora común.

**Observación:** Se realiza la reunión con el número mínimo de participantes. Además se informa a los invitados de la gestión de reuniones. Se comunica también a la Ingeniera Alexandra Corral que no pudo asistir porque no dispone de hora común.

**4.5.2.- CASO 2**

**Gestión reuniones:** Convocar reunión

**Descripción:** Reunión

**Lugar:** Departamento de Electrónica

**Prioridad:** Alta

**Fecha reunión:** 2008-07-22

**Hora reunión:** 14:30

**Tiempo de duración:** 01:00 hora

**Número mínimo de participantes:** 2

**Convocante:** Ingeniero José Luis Carrillo

**Invitados:** Ingenieros: Edison Espinosa, Armando Álvarez

<b>ING. JOSÉ LUIS CARRILLO</b>		<b>ING. EDISON ESPINOSA</b>	
<b>22-7-2008</b>		<b>22-7-2008</b>	
<b>HORARIO</b>	<b>MARTES</b>	<b>HORARIO</b>	<b>MARTES</b>
07:00 - 07:15		07:00 - 07:15	
07:15 - 09:15	Computación II	07:15 - 09:15	Computación
09:15 - 09:30		09:15 - 14:00	
09:30 - 11:30	Computación II	14:00 - 14:45	Redactar documentos
11:30 - 12:00		14:45 - 15:30	<b>REUNION</b>
12:00 - 14:00	Computación II	15:30 - 18:00	
14:00 - 14:30		18:00 - 20:00	Defensa de tesis
14:30 - 15:00	<b>REUNION</b>	20:00 - 21:30	
	Revisar capítulos tesis		
15:00 - 16:00			
16:00 - 21:30			

**Tabla 4.8 Horarios de los Ing. José Luis Carrillo y Edison Espinosa del día 22-07-2008**

<b>ING. ARMANDO ÁLVAREZ</b>	
<b>22-7-2008</b>	
<b>HORARIO</b>	<b>MARTES</b>
07:00 - 07:15	
07:15 - 09:15	Procesamiento Digitales de señales
09:15 - 09:30	
09:30 - 11:30	Procesos Estocásticos y Filtraje lineal
11:30 - 12:00	
12:00 - 13:00	Gestionar documentos
13:00 - 14:30	
14:30 - 15:30	<b>REUNION</b>
15:30 - 16:00	
16:00 - 16:45	Control equipos
16:45 - 17:00	
17:00 - 18:00	Sistemas Lineales
18:00 - 21:30	

**Tabla 4.9 Horario del Ing. Armando Álvarez del día 22-07-2008**

**Respuesta:** Se ha convocado una nueva reunión el: 2008-07-22 a las: 14:30.

**Observación:** Se convoca la reunión a la hora establecida, pero se cancela la reunión del caso 1, debido a que hay un cruce entre los rangos de hora de éstas dos reuniones y además éste convocante tiene mayor prioridad que el de la reunión anterior. Se informa de la nueva reunión y de la cancelación del caso 1 a los respectivos participantes.

#### 4.5.3.- CASO 3

**Gestión reuniones:** Convocar reunión

**Descripción:** Reunión

**Lugar:** Rectorado

**Prioridad:** Baja

**Fecha mínima:** 2008-07-23

**Fecha máxima:** 2008-07-23

**Tiempo de duración:** 01:15 horas

**Número mínimo de participantes:** 2

**Convocante:** Rector

**Invitados:** Ingenieros: José Luis Carrillo y Armando Álvarez (Docentes del la ESPE-L)

RECTOR 23-7-2008		ING. JOSÉ LUIS CARRILLO 23-7-2008	
HORARIO	MARTES	HORARIO	MARTES
07:00 - 07:30		07:00 - 09:30	
07:30 - 12:00	Viaje a quito	09:30 - 11:30	Programación
12:00 -14:00		11:30 - 12:00	
14:00 - 15:30	Entrevista	12:00 - 14:00	Computación II
15:03 -21:30		14:00 - 15:00	
		15:00 - 17:00	Defensa de tesis
		17:00 - 17:15	
		17:15 - 19:15	Programación Orientada a Objetos
		19:15 - 20:00	
		20:00 - 21:00	Hacer presupuesto nueva carrera
		21:00 - 21:30	

**Tabla 4.10 Horario del Rector de la Institución del día 23-07-2008**



**ING. ARMANDO ÁLVAREZ**

23-7-2008	
HORARIO	MARTES
07:00 - 07:15	
07:15 - 09:15	Procesos Estocásticos y Filtraje Lineal
09:15 - 09:30	
09:30 - 14:00	Viaje a quito
14:00 - 14:30	
14:30 - 16:00	Contactar auspiciantes
16:00 - 17:00	
17:00 - 19:00	Sistemas lineales
19:00 - 21:30	

**Tabla 4.11 Horario del Ing. Armando Álvarez del día 23-07-2008**

**Respuesta:** No se puede convocar la reunión porque no existe una hora común entre los participantes. Por favor intente con otra fecha.

#### **4.5.4.- CASO 4**

**Gestión reuniones:** Convocar reunión

**Descripción:** Reunión

**Lugar:** Rectorado

**Prioridad:** Baja

**Fecha mínima:** 2008-07-24

**Fecha máxima:** 2008-07-24

**Tiempo de duración:** 01:00 horas

**Número mínimo de participantes:** 2

**Convocante:** Rector

**Invitados:** Ingenieros: José Luis Carrillo y Armando Álvarez (Docentes del la ESPE-L)

<b>RECTOR</b>		<b>ING. JOSÉ LUIS CARRILLO</b>	
<b>24-7-2008</b>		<b>24-7-2008</b>	
<b>HORARIO</b>	<b>MARTES</b>	<b>HORARIO</b>	<b>MARTES</b>
07:00 - 08:00		07:00 - 07:15	
08:00 - 09:00	Revisar documentos	07:15 - 09:15	Computación II
09:00 - 09:15		09:15 - 10:30	
10:30 - 11:30	<b>REUNION</b>	10:30 - 11:30	<b>REUNION</b>
11:30 - 12:00		11:30 - 12:00	
12:00 -13:00	Contactar a las Universidades	12:00 - 14:00	Computación II
13:00 -15:00		14:00 - 15:00	
15:00 -18:00	Viaje a Ambato	15:00 - 17:00	Programación
18:00 - 21:30		17:30 - 19:00	Redactar documentos
		19:00 - 19:30	
		19:30 - 21:00	Programación

**Tabla 4.12 Horarios del Rector de la Institución y del Ing. José Luis Carrillo del día 24-07-2008**

<b>ING. ARMANDO ÁLVAREZ</b>	
<b>24-7-2008</b>	
<b>HORARIO</b>	<b>MARTES</b>
07:00 - 07:15	
07:15 - 09:15	Procesamiento Digital de señales
09:15 - 09:30	
09:30 - 09:45	Corregir pruebas
09:45 - 10:30	
10:30 - 11:30	<b>REUNION</b>
11:30 -13:00	
13:00 -15:00	Visita a colegios
15:00 - 17:00	
17:00 - 19:00	Tutoría
19:00 - 21:30	

**Tabla 4.13 Horario del Ing. Armando Álvarez del día 24-07-2008**

**Respuesta:** Se ha convocado una nueva reunión para el: 2008-07-24 a las: 10:30.

**Observación:** Se convoca la reunión a la hora establecida, y se informa a cada participante sobre la misma.

#### **4.5.5.- CASO 5**

**Gestión reuniones:** Convocar reunión

**Descripción:** Reunión

**Lugar:** Rectorado.

**Prioridad:** Alta

**Fecha reunión:** 2008-07-24

**Hora reunión:** 14:45

**Tiempo de duración:** 01:00 hora

**Número mínimo de participantes:** 3

**Convocante:** Subdirector

**Invitados:** Ingenieros: José Luis Carrillo, Edison Espinosa, Armando Álvarez

**Respuesta:** No se puede realizar la reunión porque el (la) Ing. Rector lo ha hecho en el mismo rango de hora con algunos de sus invitados: Carrillo Medina José Luis.

**Observación:** No se convoca la reunión porque hay un cruce de horarios, no puede asistir uno de los participantes y además el Rector tiene mayor prioridad que el Subdirector.

#### **4.5.6.- CASO 6**

**Gestión reuniones:** Convocar reunión

**Descripción:** Reunión

**Lugar:** Rectorado

**Prioridad:** Alta

**Fecha reunión:** 2008-07-24

**Hora reunión:** 10:30

**Tiempo de duración:** 01:00 hora

**Número mínimo de participantes:** 2

**Convocante:** Subdirector

**Invitados:** Ingenieros: Edison Espinosa, Subdirector de Docencia

SUBDIRECTOR		ING. EDISON ESPINOSA	
24-7-2008		24-7-2008	
HORARIO	MARTES	HORARIO	MARTES
07:00 - 07:30		07:00 - 09:30	
07:30 - 10:00	Curso Capacitación	09:30 - 11:30	Computación I
10:00 - 10:30		11:30 - 12:00	
10:30 - 11:30	<b>REUNION</b>	12:00 - 14:00	Computación
11:30 - 12:30	Reunión con alumnos	14:00 - 15:00	
12:30 - 15:00			Contactar empresas para pasantías
15:00 - 17:00	Organizar festividades	15:00 - 16:00	
17:00 - 21:30		16:00 - 17:15	
			Metodología de Desarrollo de Sistemas
		17:15 - 19:15	
		19:15 - 19:30	
		19:30 - 20:00	Redactar oficio
		20:00 - 21:30	

**Tabla 4.14 Horarios del Subdirector de la Institución y del Ing. Edison Espinosa del día 24-07-2008**

SUBDIRECTOR DE DOCENCIA	
24-7-2008	
HORARIO	MARTES
07:00 - 07:30	
07:30 - 10:00	Revisar documentos
10:00 - 10:30	
10:30 - 12:30	Visita a alumnos
12:30 - 15:00	
15:00 -16:00	Estudio de las nuevas carreras
16:00 - 21:30	

**Tabla 4.15 Horario del Subdirector de Docencia del día 24-07-2008**

**Respuesta:** Se ha convocado una nueva reunión el 2008-07-24 a las 10:30.

**Observación:** Se convoca la reunión a la hora establecida, ya que no hay cruce de horarios con la reunión del caso anterior, y se informa a cada participante sobre la misma.

#### 4.5.7.- CASO 7

**Gestión reuniones:** Convocar reunión

**Descripción:** Reunión

**Lugar:** Departamento de Electrónica

**Prioridad:** Baja

**Fecha mínima:** 2008-07-23

**Fecha máxima:** 2008-07-23

**Tiempo de duración:** 01:00 horas

**Número mínimo de participantes:** 1

**Convocante:** Ing. José Luis Carrillo

**Invitados:** Ingenieros: Edison Espinosa, Raúl Cajas, Raúl Rosero

<b>ING. JOSÉ LUIS CARRILLO</b>		<b>ING. EDISON ESPINOSA</b>	
<b>23-7-2008</b>		<b>23-7-2008</b>	
<b>HORARIO</b>	<b>MARTES</b>	<b>HORARIO</b>	<b>MARTES</b>
07:00 - 08:00		07:00 - 07:15	
08:00 - 09:00	<b>REUNION</b>	07:15 - 08:00	Realizar gestiones
09:00 - 09:30		08:00 - 09:00	<b>REUNION</b>
09:30 - 11:30	Programación	12:00 - 14:00	Computación
11:30 - 12:00		14:00 - 15:00	
12:00 - 14:00	Computación II	15:00 -17:00	Gestión de Procesos y Calidad
14:00 - 15:00		17:00 - 17:15	
15:00 - 17:00	Defensa de tesis	17:15 - 19:15	Metodología de Desarrollo de sistemas
17:00 - 17:15		19:15 - 19:30	
17:15 - 19:15	Programación Orientada a Objetos	19:30 - 20:30	Revisar documentos
19:15 - 20:00		20:30 - 21:30	
20:00 - 21:00	Hacer presupuesto nueva carrera		
21:00 - 21:30			

**Tabla 4.16 Horarios del los Ings. José Luis Carrillo y Edison Espinosa del día 23-07-2008**

<b>ING. RAÚL CAJAS</b>		<b>ING. RAÚL ROSERO</b>	
<b>23-7-2008</b>		<b>23-7-2008</b>	
<b>HORARIO</b>	<b>MARTES</b>	<b>HORARIO</b>	<b>MARTES</b>
07:00 - 07:15		07:00 - 08:00	
07:15 - 09:15	Computación I	08:00 - 09:00	<b>REUNION</b>
09:15 - 09:30		09:00 - 12:00	
09:30 - 11:30	Programación	12:00 - 14:00	Computación
11:30 - 15:00		14:00 - 15:00	
15:00 - 17:00	Aplicación de BD	15:00 - 17:00	Aplicaciones Distribuidas II
17:00 - 17:15		17:00 - 17:30	
17:15 - 19:15	Programación	17:30 - 18:30	Contactar docente
19:15 - 20:00		18:30 - 19:30	
20:00 - 21:00	Preparar clases	10:39 - 21:30	Computación
21:00 - 21:30			

**Tabla 4.17 Horarios del los Ings. Raúl Cajas y Raúl Rosero del día 23-07-2008**

**Respuesta:** Se ha convocado una nueva reunión el: 2008-07-23 a las: 08:00. El Ingeniero (a), Ingenieros (as) Cajas Barba Raúl Patricio no puede asistir porque no dispone de una hora común.

**Observación:** Se convoca la reunión a la hora establecida, y se informa a los participantes que pueden asistir a la misma.

#### **4.5.8.- CASO 8**

**Gestión reuniones:** Modificar lugar (de la reunión del caso 7)

**Lugar:** Departamento de Organización y Sistemas

**Respuesta:** Se ha modificado el lugar de la reunión.

**Observación:** Se informa a los involucrados en esta reunión que el lugar ha sido modificado

#### **4.5.9.- CASO 9**

**Gestión reuniones:** Cancelar participación (de la reunión del caso 7)

**Participante:** Ing. Raúl Rosero

**Respuesta:** La reunión ha sido cancelada porque no existe el número mínimo de participantes para realizarla.

**Observación:** Se elimina el participante de la reunión ya que aún se dispone del número mínimo para realizar la misma

#### **4.5.10.- CASO 10**

**Gestión reuniones:** Cancelar participación (de la reunión del caso 7)

**Participante:** Ing. Edison Espinosa

**Respuesta:** Se ha cancelado su participación

**Observación:** Se elimina la reunión y se informa a los involucrados que ha sido cancelada por motivo de falta de participantes

# CAPITULO V

## V.- CONCLUSIONES Y RECOMENDACIONES

### 5.1.- CONCLUSIONES

- Los Sistemas Multiagentes (SMA) tratan la coordinación inteligente entre una colección de agentes autónomos, permitiendo coordinar sus conocimientos, metas, propiedades y planes para tomar una decisión o resolver un problema para los que hay múltiples métodos de resolución y/o múltiples entidades capaces de trabajar conjuntamente para solucionarlos.
- La construcción de SMA integra tecnologías de distintas áreas de conocimiento: técnicas de ingeniería del software para estructurar el proceso de desarrollo; técnicas de inteligencia artificial para dotar a los programas de capacidad para tratar situaciones imprevistas y tomar decisiones, y programación concurrente y distribuida para tratar la coordinación de tareas ejecutadas en diferentes máquinas bajo diferentes políticas de planificación.
- En la implementación del sistema es posible la integración entre aplicaciones desarrolladas en diferentes tecnologías orientadas a la web, ya que por medio de HTML (*HyperText Markup Language*), combinado con lenguajes de programación (Microsoft Visual Studio .NET, PHP, JAVA) permiten tener un estándar para el desarrollo de páginas web independiente del lenguaje utilizado.
- El uso de una ontología en un SMA es indispensable para representar el conocimiento o información que se desea comunicar o compartir entre los agentes. La herramienta utilizada para generar el código fuente de las clases java que conforman la ontología fue Protégé juntamente con el componente Beangenerator. Esta herramienta es de gran aporte porque durante el proceso de

desarrollo de los agentes se realizan adiciones o modificaciones del conocimiento manipulado, lo que permitió ahorrar tiempo en su desarrollo.

- La metodología INGENIAS tiene un enfoque más orientado a agentes. Su proceso de desarrollo es robusto, detallado a un nivel que a veces puede ser excesivo. Su uso mediante una herramienta de modelado visual llamado INGENIAS IDE facilita, pero se sigue requiriendo que el desarrollador revise la documentación del análisis y diseño del sistema para entender qué hace cada entidad y cuál es el propósito de cada relación.
- La plataforma de desarrollo JADE provee los servicios necesarios para la creación, comunicación, depuración de los agentes, facilitando en gran manera la construcción de los SMA.
- La mayor parte de la tecnología utilizada en el desarrollo del software de esta tesis son de libre distribución siendo una ventaja en la parte económica, ya que no fue necesario disponer de licencias para la utilización de herramientas que son muy costosas.
- El desarrollo de este proyecto ha permitido adquirir conocimientos de nuevas tecnologías desconocidas en nuestro medio y que son la base para el desarrollo de futuras aplicaciones.

## **5.2.- RECOMENDACIONES**

- Para la construcción de los agentes se recomienda utilizar el IDE Eclipse con el parche de aplicaciones web J2EE (Java 2 Enterprise Edition) debido a que permite adjuntar fácilmente al proyecto las librerías propias de JADE (http.jar, iopp.jar, jade.jar y jadeTools.jar).
- Para el desarrollo de un SMA se recomienda seleccionar una metodología que vaya acorde a la línea de investigación de los agentes que se van a utilizar, ya



que existen agentes basados en conocimiento, procesos concurrentes, agentes BDI (Belief, Desire, Intention), agentes como entidades computacionales que integran diferentes tecnologías, agentes que actúan en el nivel de conocimiento y que se basan en el principio de racionalidad (un agente emprende acciones porque está buscando satisfacer un objetivo). Para modelar las características de éste último tipo de agentes la metodología INGENIAS es la mejor opción.

- Como trabajo futuro para mejorar la **Agenda Multiagente para la Gestión de Reuniones (AMulGeR)** desarrollada en este proyecto, se recomienda dotar a los agentes de un comportamiento más proactivo para elegir la fecha y hora de una reunión de acuerdo a nuevas exigencias, tomar decisiones a raíz de una base de conocimiento (integración con la tecnología JESS para sistemas expertos), capturar automáticamente preferencias de un usuario, etc.
- Se recomienda a los docentes de la Carrera de Sistemas e Informática reforzar los conocimientos del lenguaje de programación JAVA a los alumnos ya que es un lenguaje muy potente, robusto, seguro y lamentablemente poco difundido en nuestro medio, desaprovechando los beneficios que ofrece.
- El área de los agentes software o inteligentes basados en el estándar FIPA conjuntamente con la plataforma JADE son pocos conocidos en nuestro país, siendo mas difundido en las naciones europeas, recomendando incentivar a la investigación de esta tecnología a los alumnos de la carrera de sistemas e informática para poder aprovechar estos conocimientos en la resolución de problemas que requieren la colaboración de algunas entidades para trabajar conjuntamente.

## WEBGRAFÍA

- V. Julin, V. Botti, Agentes Inteligentes: El Siguiete paso, en la Inteligencia Artificial, 2000.  
<http://www.ati.es/novatica/2000/145/vjulia-145.pdf>
- Alejandro Quintero, Sandra Rueda Rodríguez, María Eugenia Ucrós, Agentes y Sistemas Multiagentes: integración de conceptos básicos  
[agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm](http://agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm)
- Henri Héctor Avancini, FraMaS: Un Framework para Sistemas Multi-agente, 2000  
[www.nmis.isti.cnr.it/avancini/FraMaS\\_Tesis.pdf](http://www.nmis.isti.cnr.it/avancini/FraMaS_Tesis.pdf)
- Ramón Rizo, Faraón Llorens, Mar Pujol, Arquitecturas y Comunicación entre agentes  
[www.dccia.ua.es/dccia/inf/asignaturas/AIW/docs/ACA.pdf](http://www.dccia.ua.es/dccia/inf/asignaturas/AIW/docs/ACA.pdf)
- José M. Molina López, Jesús García Herrero y Ana M<sup>a</sup> Bernardos Barbolla, Agentes y Sistemas Multiagentes, 2004  
[www.ceditec.etsit.upm.es/index.php/Descargar-documento/3-Agentes-y-Sistemas-Multiagente.html](http://www.ceditec.etsit.upm.es/index.php/Descargar-documento/3-Agentes-y-Sistemas-Multiagente.html)
- Juan Manuel Corchado, Modelos y Arquitecturas de Agente  
[postgrado.usal.es/bisite/images/stories/publicaciones/otras/2004/c1.pdf](http://postgrado.usal.es/bisite/images/stories/publicaciones/otras/2004/c1.pdf)
- David Camacho Fernández, Sistemas Multi-agentes & CSCW  
[www.ii.uam.es/~rcobos/teaching/esp/groupware/SMA-colaborativos1.pdf](http://www.ii.uam.es/~rcobos/teaching/esp/groupware/SMA-colaborativos1.pdf)
- Juan Pavón Mestras, Agentes Inteligentes: Comunicación entre agentes  
[www.fdi.ucm.es/profesor/jpavon/doctorado/acl.pdf](http://www.fdi.ucm.es/profesor/jpavon/doctorado/acl.pdf)
- Juan Pavón Mestras, Estandarización de Sistemas de agentes  
[grasia.fdi.ucm.es/jpavon/agentes/estandares.pdf](http://grasia.fdi.ucm.es/jpavon/agentes/estandares.pdf)
- M. Amor, L. Fuentes, M. Pinto, Interoperabilidad entre Plataformas de Agentes FIPA: Una Aproximación Basada en Componentes  
[cita2003.fing.edu.uy/articulosvf/95.pdf](http://cita2003.fing.edu.uy/articulosvf/95.pdf)
- Divina Ferreiro Barreiro, Estándar FIPA “Foundation for Intelligent Physical Agents  
[trevinca.ei.uvigo.es/~pcuesta/sm/alumnos2002/FIPA.pdf](http://trevinca.ei.uvigo.es/~pcuesta/sm/alumnos2002/FIPA.pdf)
- Agentes Taxonomía y Aplicaciones

<http://www.it.uc3m.es/liliana/paginas/masinfo/agentes.doc>

- Comunicación entre Agentes  
[www.sia.eui.upm.es/grupos/ACL.pdf](http://www.sia.eui.upm.es/grupos/ACL.pdf)
- FIPA: [www.fipa.org](http://www.fipa.org)
- Web Oficial Metodología INGENIAS  
<http://grasia.fdi.ucm.es/ingenias/Spain/index.php>
- Jorge J. Gómez Sanz, Modelado de Sistemas Multi-agente, 2002  
<http://www.fdi.ucm.es/profesor/jpavon/doctorado/practicas/index.html>
- Juan Fco. Garamendi Bragado, Agentes Inteligentes: JADE, 2004  
[www.upv.es/sma/practicas/jade/Introducci%F3nJADE.pdf](http://www.upv.es/sma/practicas/jade/Introducci%F3nJADE.pdf)
- Juan A. Botía Blaya, Tutorial Básico de JADE, 2005  
[galahad.plg.inf.uc3m.es/~docweb/swagent/jade\\_texto.pdf](http://galahad.plg.inf.uc3m.es/~docweb/swagent/jade_texto.pdf)
- Juan A. Botía Blaya, Construcción de Agentes Software con JAVA + JADE  
[ants.dif.um.es/~juanbot/page\\_files/uc3m2005.pdf](http://ants.dif.um.es/~juanbot/page_files/uc3m2005.pdf)
- Juan A. Botía Blaya, JADE: Escuela de Primavera de Agentes, 2005  
[ants.dif.um.es/~juanbot/page\\_files/escuelaAgentes2005jade.pdf](http://ants.dif.um.es/~juanbot/page_files/escuelaAgentes2005jade.pdf)
- Marco de trabajo y entorno de desarrollo de agentes, basado en la exposición de Fabio Bellifemine, Telecom Italia Lab – Torino (Italy), 2001  
[www.infor.uva.es/~cllamas/MAS/IntroJADE.pdf](http://www.infor.uva.es/~cllamas/MAS/IntroJADE.pdf)
- Sitio Web JADE: <http://jade.tilab.com/>
- Programación JADE  
<http://programacionjade.wikispaces.com>
- Fipa vs JADE  
[www.dsic.upv.es/users/ia/sma/tools/jgomas/archivos/documentation/SIN\\_FIPA-JADE\\_2006-2007.pdf](http://www.dsic.upv.es/users/ia/sma/tools/jgomas/archivos/documentation/SIN_FIPA-JADE_2006-2007.pdf)
- La plataforma de agentes JADE.  
[ants.dif.um.es/~juanbot/page\\_files/escuelaAgentes2005jade.pdf](http://ants.dif.um.es/~juanbot/page_files/escuelaAgentes2005jade.pdf)
- JADE  
[www.dsic.upv.es/users/ia/sma/tools/jgomas/archivos/documentation/SIN\\_JADE\\_2000\\_sesion\\_1.pdf](http://www.dsic.upv.es/users/ia/sma/tools/jgomas/archivos/documentation/SIN_JADE_2000_sesion_1.pdf)

## ANEXO A

### INTEGRACIÓN DE JADE CON SERVLET (CLASE JADEGATEWAY)

En este anexo se explica brevemente qué es un servlet y como se comunica con la plataforma JADE por medio de la clase JadeGateway.<sup>47</sup>

#### 1.1.- INTRODUCCIÓN A LOS SERVLETS

Los servlets son objetos que corren dentro del contexto de un contenedor de servlets (ejemplo Tomcat) y extienden su funcionalidad. La palabra *servlet* deriva de otra anterior, *applet*, que se refería a pequeños programas escritos en Java que se ejecutan en el contexto de un navegador Web. Por contraposición, un *servlet* es un programa que se ejecuta en un servidor.

Un *servlet* es un objeto Java que implementa la interfaz javax.servlet.Servlet. Al implementar esta interfaz el servlet es capaz de interpretar los objetos de tipo HttpServletRequest y HttpServletResponse quienes contienen la información de la página que invocó al *servlet*.

La interface Servlet declara los métodos que se debe implementar:

void	<b>destroy()</b> Es llamado por el contenedor del servlet para finalizar el servlet.
ServletConfig	<b>getServletConfig()</b> Devuelve un objeto ServletConfig que contiene los parámetros de inicialización del sevlet

<sup>47</sup> Tutorial de Agentes: <http://programacionjade.wikispaces.com/Ontolog%C3%ADas>

String	<b>getServletInfo()</b> Devuelve información del servlet como el autor, versión ...etc.
void	<b>init(ServletConfig config)</b> Es llamado por el contenedor de servlets para poner en funcionamiento un servlet
void	<b>service(ServletRequest req, ServletResponse res)</b> Es llamado por el contenedor de servlets para permitir al servlet contestar a una petición.

**Tabla a.1: Métodos de un servlet**

En este proyecto se utilizó la clase `javax.servlet.HttpServlet` que permite sobrescribir los métodos necesarios para la correcta inicialización, destrucción y proceso de peticiones tanto get como post desde un formulario de la página PHP. Los métodos son los siguientes:

- `doGet`, para el proceso de peticiones get
- `doPost`, para el proceso de peticiones post
- `doPut`, para el proceso de peticiones put
- `doDelete`, para el proceso de peticiones delete
- `init and destroy`, manejo de recursos que serán útiles en la vida del servlet
- `getServletInfo`, provee información sobre el servlet como autor, versión.

## 1.2.- INTRODUCCIÓN AL PAQUETE `jade.wrapper.gateway`

El paquete principal de Jade que se utilizó para la comunicación con el servlet es `jade.wrapper.gateway`, que incluye las clases: `JadeGateway`, `GatewayAgent`, `GatewayBehaviour`

La clase `JadeGateway` provee una pasarela para poder conectar código no JADE con sistemas multiagente basados en JADE. Esta pasarela mantendrá un agente (de la clase `GatewayAgent`) que será el que se comunique directamente con el servlet. La

activación/terminación de este agente se gestiona completamente mediante la clase JadeGateway, no hace falta que el programador se preocupe por ello.

JadeGateway permite dos formas de implementar la pasarela para comunicación:

La primera (y la que se usó en este proyecto) es crear un agente que derive de GatewayAgent. De esta forma se implementó dos métodos:

- *processCommand*: Establece la comunicación, recibiendo por parámetro un objeto que contiene la información que se necesita para realizar las operaciones oportunas.
- *releaseCommand*: Una vez que se haya completado las operaciones a realizar por el sistema multiagente, se devolverá el objeto al servlet, con la información de respuesta que se estime oportuna. Este ciclo de comunicación podrá repetirse hasta que terminemos el agente de pasarela que hemos creado con JadeGateway.shutdown().

La segunda consiste en implementar un comportamiento que derive de la clase GatewayBehaviour, para que un agente propio (sin que tenga que derivar de GatewayAgent) pueda funcionar como pasarela. Para ello se añade al agente una instancia de un comportamiento que derive de GatewayBehaviour. Este comportamiento también tendrá que implementar los métodos, processCommand y releaseCommand.

El agente que actúe como pasarela, será iniciado por medio de la clase JadeGateway desde el servlet, por medio del método JadeGateway.init(). Este método recibe como parámetros el agente (el nombre de la clase que implementa el agente, no el nombre del agente en sí) y una serie de propiedades que definirán parámetros como el Host y el Puerto en el que se está ejecutando la plataforma Jade con la que se quiere comunicar. Una vez iniciado, se realiza la comunicación por medio del método JadeGateway.execute(), al que se le pasa el objeto que se recibe en el agente pasarela por medio de processCommand().

### 1.3.- INTEGRACIÓN DE JADE CON SERVLET

A continuación se presenta un ejemplo simple para poder entender su funcionamiento.

Se tiene una página html para enviar un formulario al servlet con una acción "enviarmensaje", el servlet es el encargado de elegir la acción adecuada que estará almacenada en una tabla hash (clave, AcciónXXX) donde acción es el tipo de acción a realizar (en este caso " enviarmensaje") y AcciónXXX será un objeto que permitirá realizar dicha acción. Este objeto AcciónXXX será el encargado de crear el mensaje a enviar por medio de la clase Mensaje.java y enviárselo al GateWayAgent. El GateWayAgent recibirá ese mensaje, construirá un mensaje ACL y se lo enviará al AgenteSaludo que le contestará con otro mensaje ACL. El GateWayAgent devuelve la contestación al Servlet (al objeto AcciónEnviarMensaje) y este la muestra en el navegador.

Los elementos del ejemplo serán los siguientes:

- index.html: es la página html encargada de mandar la acción "enviarmensaje" al servlet.
- ServletGateWay.java: es el encargado de procesar la acción a realizar y devolver una respuesta que le llegará desde JADE
- Acción.java: interfaz genérica para todas las posibles acciones del servlet, define un método enviar de tal forma que todas las acciones sean invocadas de la siguiente forma: AcciónXXX.enviar(...);
- AccionEnviarMensaje.java: clase que crea un mensaje predeterminado y lo envía y muestra la respuesta de JADE.
- MiAgenteGateWay.java: clase que deriva de GateWayAgent y sirve de pasarela entre el servlet y JADE. Además se encarga de enviar un mensaje ACL al AgenteSaludo y procesar su contestación enviándola de vuelta al Servlet.
- AgenteSaludo: clase que implementa un agente que procesa un mensaje ACL y devuelve una contestación predefinida.

A continuación se presenta una figura del pequeño ejemplo con los pasos más importantes a realizar:

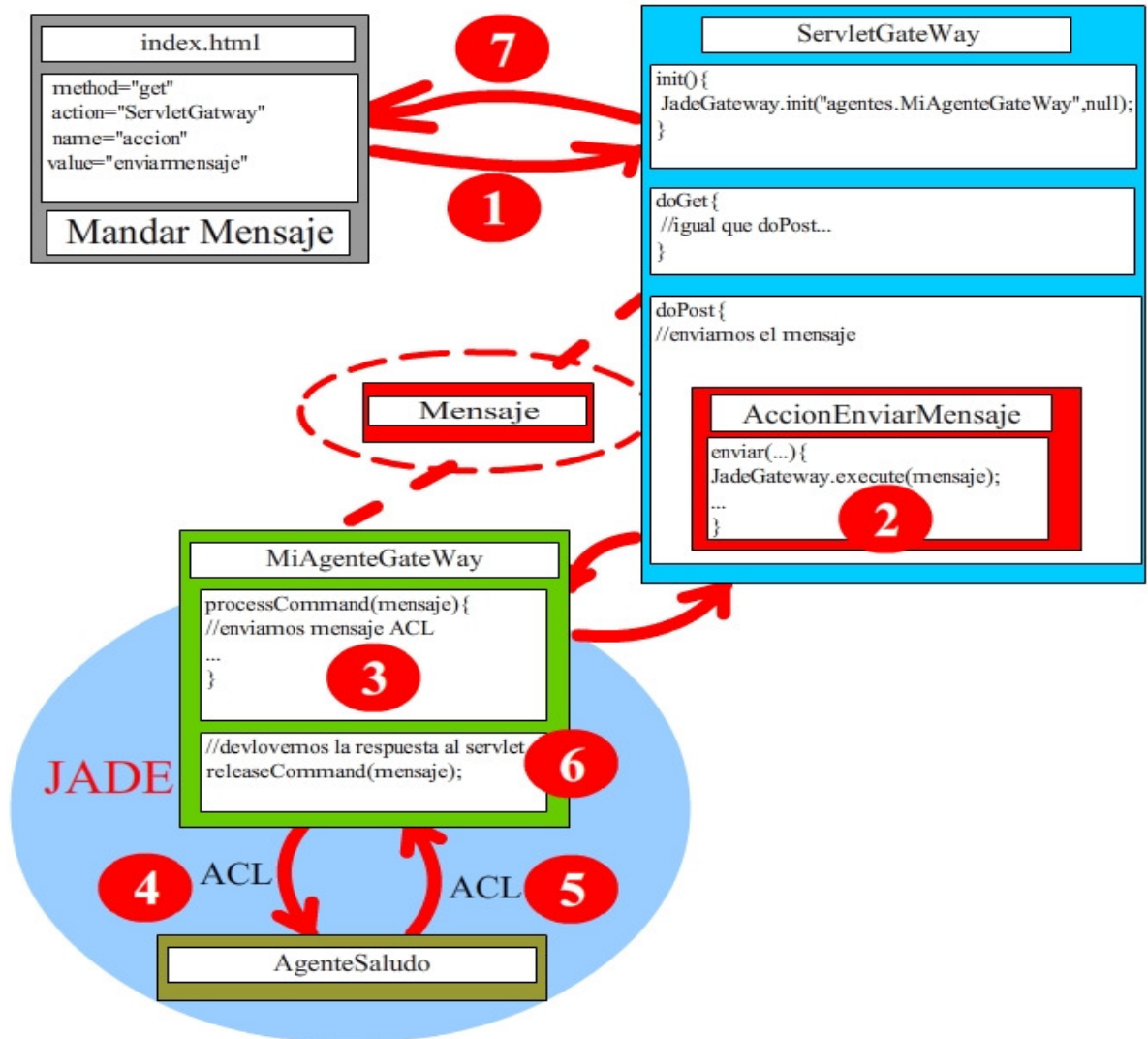


Figura a.1 Pasos a seguir en la integración de JADE con Servlet

A continuación se presenta el código de las respectivas clases y página html.



## index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>Ejemplo de uso de JadeGateWay</title>
  </head>
  <body id="page">
    <br><h1>Ejemplo1 de uso de JadeGateWay</h1><br>
    <HR>
    <form id="operationBox" method="post" action="ServletGateWay">
      <input type="hidden" name="accion" value="enviarmensaje"/>
      <input type="submit" value="Mandar Mensaje">
    </form>
  </body>
</html>
```

## ServletGateWay.java

```
package servlet;

/*****

Servlet que recibe las peticiones del usuario o de la página web

*****/

import jade.core.Profile;
import jade.core.ProfileImpl;
import jade.core.Runtime;
import jade.wrapper.AgentController;
import jade.wrapper.ContainerController;
import jade.wrapper.gateway.JadeGateway;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import enviarmensaje.*;
import mensaje.Mensaje;

import java.util.*;
```

```

import jade.util.leap.Properties;

public class ServletGateWay extends HttpServlet
{
    static final long serialVersionUID = 1L;
    Hashtable acciones = null;

    // las peticiones get son tratadas como las post
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {

        doPost(request,response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        // se obtiene el valor del campo oculto accion
        String nombreAccion = request.getParameter("accion");

        if (nombreAccion == null)
        {
            response.sendError(HttpServletResponse.SC_NOT_ACCEPTABLE);
            return;
        }

        //se hace que el objeto implemente la interfaz acción
        //para que pueda utilizar los objetos HttpServletRequest y HttpServletResponse
        Accion accion = (Accion) acciones.get(nombreAccion);
        if (accion == null)
        {
            response.sendError(HttpServletResponse.SC_NOT_IMPLEMENTED);
            return;
        }

        // se ejecuta la acción
        accion.enviar(this, request, response);
    }

    public void init() throws ServletException
    {

        //creación de la tabla hash que contendrá el objeto AccionEnviarMensaje
        acciones = new Hashtable();
    }
}

```

```

// se inserta el objeto con clave su clave en la tabla
acciones.put("enviarmensaje",new AccionEnviarMensaje());

//se enlaza con el agente GateWay de JADE
JadeGateway.init("agentes.MiAgenteGateWay",null);

}

}

```

### **Accion.java**

```

package enviarmensaje;

/*****

Interface genérica para el servlet

*****/
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public interface Accion
{

    public void enviar(HttpServletRequest servlet, HttpServletRequest request,
    HttpServletResponse response)
        throws IOException, ServletException;

}

```

### **AccionEnviarMensaje.java**

```

package enviarmensaje;

/*****

La clase AccionEnviarMensaje sale fuera del servlet y envía el mensaje al AgenteGateWay

*****/

import jade.wrapper.gateway.JadeGateway;

import java.io.IOException;
import java.io.PrintWriter;

```

```

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import mensaje.Mensaje;

public class AccionEnviarMensaje implements Accion
{

    public void enviar(HttpServlet servlet, HttpServletRequest request, HttpServletResponse
response)
    throws IOException, ServletException
    {

        // se crea un mensaje con un destinatario y contenido predefinido
        Mensaje mensaje = new Mensaje();
        mensaje.setReceiver("AgenteSaludo");
        mensaje.setMessage("Hola querido amigo, ¿cómo estás?");

        try {
            // se accede a JADE via JadeGateWay y se espera la contestación
            JadeGateway.execute(mensaje);
        } catch(Exception e) { e.printStackTrace(); }

        // se crea la salida
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        //se imprime la contestación recibida de JADE
        out.print("¡El mensaje ha sido enviado!<br/>");
        out.print("Contestación:"+mensaje.getMessage()+"<br/>");
        out.print("<br/><a href='index.html'> Volver a enviar </a>");

        out.flush();
        out.close();

    }

}

```

### **Mensaje.java**

```

package mensaje;

/*****

```

La clase Mensaje es el canal de comunicación entre AgenteGateWay y el ServletGateWay

```
*****/
public class Mensaje implements java.io.Serializable
{
    private String message = new String("");
    private String receiver = new String("");

    public String getMessage(){
        return message;
    }

    public void setMessage(String str){
        message=str;
    }

    public String getReceiver(){
        return receiver;
    }

    public void setReceiver(String receiver){
        this.receiver=receiver;
    }

    public Mensaje(){
    }
}

```

### **MiAgenteGateWay.java**

```
package agentes;

/*****/

Este agente recibe el mensaje y se lo envía al agente AgenteSaludo

*****/

import mensaje.Mensaje;

import jade.core.AID;
import jade.core.behaviours.OneShotBehaviour;
import jade.core.behaviours.CyclicBehaviour;
import jade.domain.AMSService;
import jade.domain.FIPAAgentManagement.AMSAgentDescription;
```

```

import jade.domain.FIPAAgentManagement.SearchConstraints;
import jade.lang.acl.ACLMessage;
import jade.wrapper.gateway.*;

public class MiAgenteGateWay extends GatewayAgent
{
    Mensaje mensaje = null;
    //método que se ejecuta cuando se invoca JadeGateWay.execute(objeto) en el servlet
    protected void processCommand(java.lang.Object obj)
    {
        if (obj instanceof Mensaje)
        {
            mensaje = (Mensaje)obj;
            ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
            msg.addReceiver(new AID( mensaje.getReceiver(), AID.ISLOCALNAME) );
            msg.setContent(mensaje.getMessage());
            send(msg);
        }
    }

    public void setup()
    {
        //Esperando por la respuesta de AgenteSaludo
        addBehaviour(new CyclicBehaviour(this)
        {
            public void action()
            {
                ACLMessage msg = receive();
                if ((msg!=null)&&(mensaje!=null))
                {
                    mensaje.setMessage(msg.getContent());
                    //se devuelve la respuesta en el objeto mensaje al servlet
                    releaseCommand(mensaje);
                } else block();
            }
        });
        super.setup();
    }
}

```

### **AgenteSaludo.java**

```

package agentes;

/*****

Agente que contesta a los mensajes que le llegan con un saludo

*****/

```

```

import jade.core.Agent;
import jade.core.behaviours.*;
import jade.lang.acl.*;

import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.*;
import jade.domain.FIPAException;

public class AgenteSaludo extends Agent
{
    protected void setup()
    {
        // Comportamiento a la espera de un mensaje
        addBehaviour(new CyclicBehaviour(this)
        {
            public void action()
            {
                ACLMessage msg = receive();
                String content= "";

                if (msg!=null) {
                    //rellenamos el contenido de la contestación
                    content=
                    "<br/> - " + myAgent.getLocalName() + " recibí: " + msg.getContent()+
                    "<br/> - " + myAgent.getLocalName() + " envió: " + "Yo estoy bien , ¿y tú?";

                    ACLMessage reply = msg.createReply();
                    reply.setPerformative( ACLMessage.INFORM );
                    reply.setContent(content);
                    send(reply);
                    System.out.print(content);
                }
                else block();
            }
        });
    }
    protected void takeDown()
    {
        try { DFService.deregister(this); }
        catch (Exception e) {}
    }
}

```

**Latacunga, Agosto del 2008**

---

**Eithel Jennifer Navarrete Carreño**  
**C.I. No. 091994833-1**

---

**Ing. Edison Espinosa**  
**COORDINADOR DE LA CARRERA DE SISTEMAS E INFORMÁTICA**

---

**Dr. Rodrigo Vaca**  
**SECRETARIO ACADÉMICO**