Diseño de protocolos de Comunicación en un Sistema de Información MultiAgente para el Soporte a la Toma de Decisiones Médicas

Hadad A[1,2], Evin D[1], Martina M[1], Drozdowicz B[1,2]
[1]Grupo de Inteligencia Artificial Aplicada, Facultad de Ingeniería, Universidad Nacional de Entre Ríos, Argentina. [2]Facultad de Ciencia y Tecnología, Universidad Autónoma de Entre Ríos, Argentina

Resumen
Una de las características fundamentales que presentan los sistemas multiagentes es la capacidad de comunicación e interacción entre cada uno de sus componentes. Tal proceso de comunicación se lleva a cabo mediante el intercambio de mensajes y, para la comprensión mutua, es vital que los agentes acuerden respecto al formato y la semántica de sus mensajes.
En este trabajo se presenta el diseño de interacciones que tienen lugar en un Sistema de Información Multiagente, cuyo objetivo es brindar soporte a la toma de decisiones en el diagnóstico temprano en procesos médicos complejos. Se propone la integración del estándar DICOM con el lenguaje de comunicación FIPA, con el fin de permitir la introducción de imágenes y señales fisiológicas en el contenido de mensajes, y al mismo tiempo facilitar la interconectividad entre diversos equipos de diagnóstico y monitoreo. Se muestran algunos avances en la implementación de esta propuesta en el marco de la herramienta de desarrollo de sistemas multiagentes JADE.

Palabras Clave: Sistema de Información, Protocolos de Comunicación, Toma de Decisiones

Introducción

En la teoría de sistemas multiagentes, ningún agente puede forzar a otros a realizar una acción determinada. Por lo general, la única alternativa que tiene un agente para lograr que otros agentes hagan algo que el primero desea, es a través de actos comunicativos.
A través de tales actos comunicativos un agente busca interactuar con sus pares con el propósito de influenciarlos adecuadamente para la concreción de su objetivo, es decir, de forma tal que producto de esa interacción logre que éstos quieran realizar las acciones solicitadas.

En 1995 la Fundación para Agentes Físicos Inteligentes (FIPA) comenzó a desarrollar estándares para la implementación de sistemas basados en agentes. La pieza central de esta iniciativa fue el desarrollo de un lenguaje de comunicación entre agentes (ACL). El ACL de FIPA define un lenguaje externo para los mensajes, y 20 performativas para definir la interpretación deseada de los mensajes. Éstas últimas corresponden a tipos diferentes de actos de diálogo que por el simple hecho de ser nombrados se convierten en acciones como: solicitar, informar, prometer, etc. Sin embargo, FIPA-ACL no especifica ningún lenguaje para el contenido del mensaje [1].

En este trabajo se discuten las interacciones entre agentes que tienen lugar en un sistema de información para el soporte a la toma de decisiones en procesos médicos complejos. Estos sistemas contemplan la utilización tanto de imágenes como de señales fisiológicas del ámbito médico. Debido a que el estándar DICOM contempla tanto la transmisión de

imágenes como señales médicas, se propone integrarlo en la especificación del contenido de mensajes FIPA-ACL.

Como antecedentes en la utilización de DICOM en sistemas multiagentes, en [2] se describe un framework denominado *Agent.Hospital* que facilita la comunicación entre diferentes sistemas multiagentes de un hospital. Se busca que cada sistema sea capaz de usar la funcionalidad de módulos HL7 o DICOM para enviar mensajes específicos, recibir confirmaciones, visualizar, procesar y enviar imágenes DICOM desde un archivo usando protocolos DICOM. Además, los primeros prototipos de los módulos HL7 y DICOM fueron diseñados para ser compatibles con agentes FIPA tal que puedan manipular órdenes de exámenes o agendar solicitudes.

En [3] se describe un sistema biomédico de aprendizaje sensible al contexto (context-aware) que contempla aprendices móviles de una red social.
Este sistema de aprendizaje electrónico denominado Bio-SAmI emplea técnicas de inteligencia en ambientes sociales y considera a los usuarios móviles en el marco de modelo de "actores". Este modelo se implementa usando una combinación de APIs basadas en Java, entre las que se incluye a Jade. Se implementa un prototipo que permite a los alumnos compartir información representada mediante el estándar DICOM en relación a la noción de inflamación, así como responder a una variedad de consultas de aprendizaje que incluyen la clasificación de casos de estudio, encontrar casos de estudios específicos, localización de alumnos FOAF (amigo de un amigo), así como la publicación y agrupación de casos de estudio.

Finalmente, en [4] se presenta un modelo basado en la integración de tecnologías Web y agentes de software para el diseño de un ambiente colaborativo para el diagnóstico auxiliado por computadoras. Este sistema permite a los médicos obtener el diagnóstico adecuado para los archivos de datos de un paciente actuando de manera cooperativa, integrando de forma dinámica agentes Jade en un entorno SOA de servicios Web. Se propone utilizar DICOM en el formato de los archivos de imágenes y un servidor Web para la distribución de los datos.

A diferencia de los antecedentes mencionados, en este trabajo se estudia la aplicación del estándar DICOM dentro de la estructura de mensajes FIPA, en un sistema de soporte de decisiones del ámbito médico. El tipo de aplicación descripta se caracteriza y diferencia de los casos antes mencionados por el uso intensivo y coordinado que supone de señales e imágenes médicas para el procesamiento y análisis inteligente de datos. Esto hace que sea un escenario particularmente rico para analizar la comunicación entre agentes que hacen uso de diversas señales e imágenes médicas.
El trabajo está organizado de la siguiente manera: en primer lugar se brinda una descripción general del sistema de información propuesto, detallando específicamente las interacciones entre agentes, y la metodología de diseño empleada. Posteriormente se describen algunas características del estándar DICOM que lo hacen pertinente para la aplicación de estudio, así como las particularidades de los protocolos de comunicación entre agentes en el marco del software Jade, elegido para la implementación del prototipo.
Finalmente se presentan algunos detalles de implementación del sistema propuesto.

Elementos del Trabajo y metodología

Gaia
Gaia constituye un intento por definir una metodología general y completa para el análisis y diseño de Sistemas MultiAgentes (SMAs). Soporta los distintos niveles presentes en el proceso de desarrollo, la estructura individual del agente y la sociedad de agentes del sistema. En Gaia el SMA esta compuesto de un número de agentes interactivos y autónomos que viven en una sociedad organizada en la que cada agente desempeña una o más funciones específicas. En Gaia se define la estructura de un SMA en términos de un "modelo de roles", el modelo identifica los roles que desarrollan los agentes para interactuar en el sistema y los protocolos de interacción entre ellos.

El objetivo del proceso de análisis en Gaia es la identificación de los roles y el modelo de interacción entre tales roles. Los roles poseen cuatro atributos: "responsibilities, permissions, activities y protocols."

- **Responsibilities**, es la propiedad que define la funcionalidad del rol, hay de dos tipos, liveness properties, que agrega alguna buena utilidad al sistema y "safety properties" que tratan de evitar peligros al sistema.
- **Permissions**, caracteriza las acciones que el rol tiene permitido hacer y las fuentes a las que tiene el acceso permitido.
- **Activities**, son las tareas que el agente debe realizar sin interactuar necesariamente con otros agentes.
- **Protocols**, son los patrones de interacción específicos, por ejemplo un rol de vendedor puede soportar varios protocolos de subasta.

Gaia además tiene operadores formales y plantillas para la representación de funciones y sus atributos; además presenta esquemas que se pueden utilizar para la representación de las interacciones entre las diversas funciones en un sistema.

En el proceso de diseño siguiendo la metodología Gaia, el primer paso es mapear los roles en tipos de agentes y crear el número adecuado de instancias de cada tipo. Un tipo de agente puede construirse con uno o más roles.
El segundo paso consiste en determinar el modelo de servicios necesarios para que un agente cumpla su rol. Un servicio puede ser visto como una función del agente y puede ser derivado de la lista de protocolos, actividades y responsabilidades de cada rol.
Finalmente el último paso es crear el modelo de interacción, que es un gráfico que representa las rutas de comunicación entre los diferentes tipos de agentes.

Plataforma JADE
JADE (Java Agent DEvelopment Framework) es básicamente un entorno de desarrollo de software para la implementación de Sistemas MultiAgentes y otras aplicaciones que cumplan con los estándares FIPA para agentes inteligentes. Posee una infraestructura flexible que permite una fácil extensión empleando módulos adicionales. Al estar escrito en Java, se beneficia del enorme conjunto de características del lenguaje y de las bibliotecas de terceros, y por lo tanto ofrece un rico conjunto de abstracciones de programación que permite a los desarrolladores construir sistemas multiagente, aún teniendo mínima experiencia en la teoría de agentes. JADE fue inicialmente desarrollado por el departamento de Investigación y Desarrollo de Telecom Italia SPA, pero ahora es un proyecto comunitario y distribuido como código abierto bajo la licencia LGPL.

JADE proporciona la tecnología estándar de soporte a agentes y ofrece a los desarrolladores una serie de características para simplificar el proceso de desarrollo:
• Plataforma de agentes distribuida. La plataforma de agentes puede estar distribuida en varias máquinas, cada una de ella ejecuta una máquina virtual Java.
• Plataforma de Agentes conforme a FIPA, que incluye al agente de gestión del sistema, el facilitador de directorio y el canal de comunicación entre agentes.
• Transporte eficiente de mensajes ACL entre agentes.
• Entorno de ejecución. Donde los agentes JADE interactúan.
• Biblioteca de clases, un conjunto de herramientas gráficas de administración y control, biblioteca de documentación, y un conjunto de librerías de ejemplos y demos.

Estándar DICOM
El estándar DICOM se puede ver como un conjunto de reglas que hacen posible el intercambio, almacenamiento y visualización de imágenes digitales.
Desde el año 2000 dicho estándar incluye además una sección especial denominada "DICOM Waveform" dedicada a señales fisiológicas como ondas electrocardiográficas (ECG). Además el estándar tiene estipulado la forma de agregar información valiosa para el diagnóstico que acompaña a las señales fisiológicas como observaciones en el trazo del electrocardiográfico, anotación de los puntos característicos de estas señales (ondas P, QRS, T), eventos especiales (como latidos ectópicos, supraventriculares), esenciales para el soporte de decisión.

Sin embargo la adopción de Dicom-Waveforms es mucho más reciente. Recién en el año 2006 se comenzaron a fabricar electrocardiógrafos comerciales con calidad diagnóstico y soporte DICOM. A través de estos equipos se puede por ejemplo enviar las señales de ECG a los servidores de la red DICOM, para analizarlas de manera conjunta con otras imágenes como angiografías cardiovasculares.

En relación al Sistema Multiagentes (SMA) propuesto, el estándar facilita la interconexión de diversos dispositivos de monitoreo, posiblemente fabricados por empresas distintas con el sistema de información.

Comunicación interagentes en JADE
El software para implementación de sistemas multiagentes JADE emplea el estándar FIPA-ACL para la comunicación entre agentes. Esto quiere decir que idealmente, un agente Jade podría interactuar con agentes FIPA escritos en otros lenguajes y corriendo bajo otras plataformas. Jade soporta el Lenguaje Semántico de FIPA, una codificación de conceptos, acciones y predicados en estilo LISP y también permite que el contenido del mensaje esté dado por objetos Java serializados.

Además del contenido, la estructura de un mensaje Jade también contiene: identidad del receptor, identidad del emisor y el tipo de mensaje. Para que un agente lo pueda entender, todas las partes del mensaje deben respetar un formato común.

A continuación se presenta una lista de atributos de un mensaje Jade FIPA-ACL. Jade brinda los métodos get y set para acceder a todos estos atributos.

* Performative - tipo de mensaje FIPA (INFORM, QUERY, PROPOSE, ...)
* Addressing
    o Receiver

o Sender (inicializado automáticamente)
* Content - Contenido principal del mensaje
* ConversationID - Usado para vincular mensajes en la misma conversación

* Language - Especifica el lenguaje que se usa en el contenido
* Ontology - Especifica la ontología que se usa en el contenido

* Protocol - Especifica el protocolo
* ReplyWith - Campo que ayuda a distinguir respuestas
* InReplyTo - Campo que usa el emisor para distinguir respuestas
* ReplyBy - Usado para establecer un límite de tiempo para una respuesta

Cuando se crea un mensaje se debe indicar su tipo (performativa), y establecer su contenido, como se muestra a continuación:

```
ACLMessage msg = new ACLMessage( ACLMessage.INFORM );
msg.setContent("Vendo calamares a $10/kg" );
```

Este mensaje usa la performativa más común: INFORM por la cual un agente brinda a otro alguna información útil. Otros tipos son: QUERY para hace una pregunta, REQUEST para solicitar a otro que haga algo, y PROPOSE para comenzar un regateo. Las performativas para respuestas incluyen AGREE o REFUSE.

RESULTADOS

Modelado del SMA con Gaia

El objetivo del Sistema MultiAgente a desarrollar era brindar soporte a la toma de decisiones durante el seguimiento del estado de pacientes en unidades de cuidados críticos. Para llegar al diseño de la arquitectura de dicho Sistema se siguió la metodología GAIA ya descripta, la cual permitió llegar a un diseño acabado a través de los modelos de agentes, modelo de conocidos y modelos de servicios a partir de los roles del sistema definidos por protocolos, permisos, actividades y responsabilidades.

Dadas las características del dominio del problema que nuestro sistema quiere atacar, se puede caracterizar el ambiente de funcionamiento del sistema como cerrado y acotado. Asimismo, el sistema tendrá un conjunto limitado de agentes cuyas habilidades, servicios y relaciones interagentes no cambiarán en tiempo de ejecución. Tales características se ajustan a las principales características de los dominios para los cuales GAIA es apropiada.

Identificación de Roles y protocolos

A pesar que GAIA afirma que permite llevar sistemáticamente al analista desde el estado de requerimientos hasta un diseño suficientemente detallado, no presenta ningún procedimiento que guíe la identificación de roles y protocolos [6, 7], razón por la cual se decidió aplicar la aproximación de Villarreal et al. (2002), que propone identificar roles y protocolos viendo al sistema como un conjunto de procesos de negocios. Dicha aproximación consiste de 3 etapas:

- **Etapa 1**: definición de la meta del sistema.
- **Etapa 2**: definición de los procesos que permitirán alcanzar dicha meta a través sus entradas, salidas y actividades.
- **Etapa 3**: identificación de los roles y protocolos necesarios para realizar las actividades anteriormente definidas.

Finalizada la tercera etapa, se construyen los Modelos de Roles, de Interacción que caracterizarán la Arquitectura del sistema.

3.1.1 Definición de la meta del sistema

Se estableció como meta del sistema:

"Reducir la posibilidad de daños en pacientes en Unidades de Cuidados Intensivos, brindando soporte al proceso de supervisión de los estados del paciente mediante diagnósticos tempranos".

3.1.2 Definición de los procesos y sus actividades

Para esta etapa la propuesta de Villarreal et al. indica modelar primeramente al sistema como un conjunto de procesos, definiendo como ya se mencionó anteriormente, sus entradas, salidas y orden de precedencia de las actividades que los conforman.
Aplicando este paso, se definió que el sistema a construir se podría considerar conformado por un solo proceso: "Dar soporte a los procesos de supervisión en Unidades de Cuidados Intensivos".

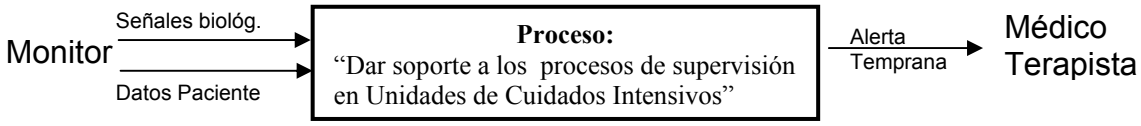


Figura 1. Entradas y salidas de los procesos.

Las entradas que recibe el proceso son las siguientes:

- **SEÑALES BIOLÓGICAS:** provienen de los monitores multiparamétricos de las unidades de cuidados intensivos. Comprenden entre otras señales electrocardiograma, presión arterial invasiva, curva pletismográfica e impedancia respiratoria (muestreados periódicamente). De estas señales se derivan parámetros de valor clínico como frecuencia cardíaca, presión arterial, frecuencia respiratoria y saturación de oxigeno. Se utilizan para que el sistema deduzca el estado actual del paciente.

- **DATOS PACIENTE**: datos no temporales que caracterizan al paciente y puedan ser de utilidad para el diagnostico clínico, como por ejemplo, edad, sexo, peso, talla, antecedentes, etc.

- **ALERTAS**: se corresponden con diagnósticos y advertencias prematuras, que permiten tomar decisiones en forma anticipada a posibles complicaciones en el estado de los pacientes.

Siguiendo la propuesta de Villarreal et al., se definió el siguiente diagrama de actividades para el proceso principal del sistema.

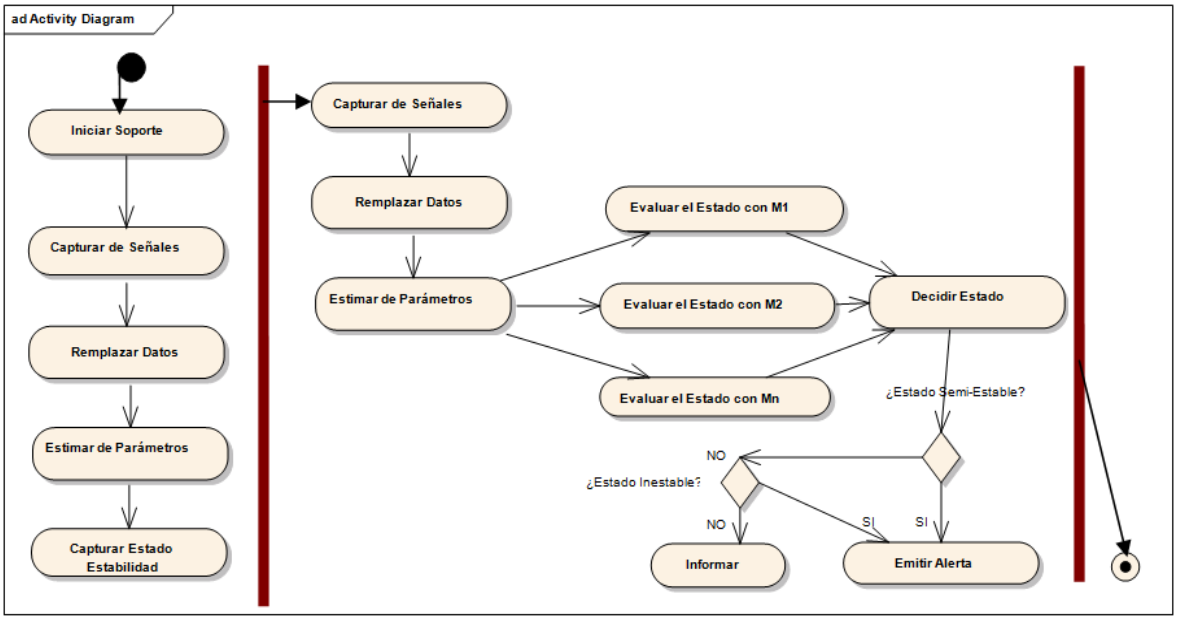


Figura 2. Diagrama de actividades del proceso.

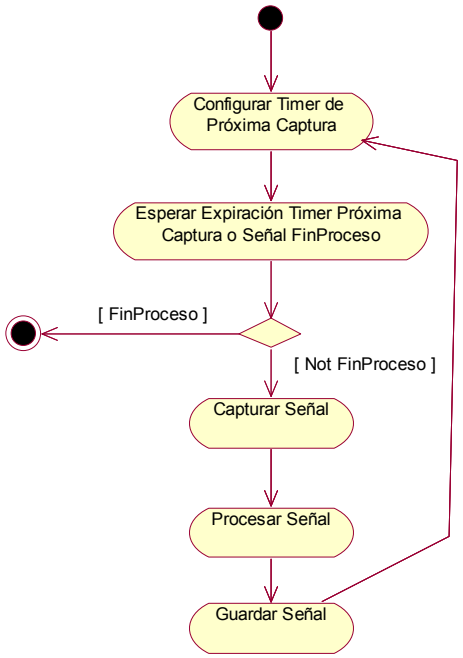
Como se puede ver, este proceso se ha dividido en dos fases: una de inicio o configuración, y la otra correspondiente a la fase de soporte propiamente dicha.
La fase inicial comprende las primeras tres actividades del proceso principal, mientras que la fase de soporte consiste en la ejecución del proceso de supervisión propiamente dicho, y estáa conformada por un conjunto de actividades, algunas de las cuales que se ejecutan en forma concurrente, para dar lugar a otras actividades que cierran un ciclo de proceso.

Durante la fase de configuración, las tres actividades permiten al sistema capturar el estado de referencia estable paciente. Durante la fase de soporte, como se mencionó anteriormente, parte de las actividades se ejecutarán de manera concurrente.

Como se podrá observar, el diagrama de la figura 2 no logra modelar el hecho de que las actividades dentro de las barras de sincronización, mientras no reciban una señal de finalización de proceso, se ejecutarán desde el principio una y otra vez formando un ciclo. Como se verá más adelante, se logra modelar esta situación analizando cada actividad en mayor detalle.

Para completar la etapa 2, Villarreal et al. proponen realizar un cuadro por cada actividad del proceso donde se detallan sus distintos atributos: objetivo, entradas, salidas, lista de tareas, restricciones y recursos.

Debido a la complejidad del flujo de control que se da lugar en varias de las actividades del proceso, se consideró conveniente realizar, en lugar de una lista de tareas como se propone originalmente, un diagrama de actividad. Esto permite representar bucles, flujos alternativos y tareas posiblemente concurrentes. A continuación se muestran los atributos de una de las actividades anteriormente mencionadas.

Actividad	Capturar Señales
Objetivo	En forma periódica, toma una muestra de las señales de los monitores y los guarda para su posterior análisis.
Entradas	Señales de los monitores.
Salidas	Señales Filtradas.
Tareas	
Restricciones	-
Recursos	Escribe Base de señales

Fase de Soporte

Actividad	Evaluar el Estado con M1
Objetivo	Se encarga de identificar estados Estable, Semi-Estables e Inestables mediante el modelo M1
Entradas	Base de Parámetros
Salidas	Estado Caracterizado con M1
Tareas	1. Leer Parámetros 2. Procesar el Modelo M1 3. Asociar el Resultado a un tipo de Estado 4. Guardar Estado Caracterizado con M1
Restricciones	
Recursos	Lee: Base de Parámetros Escribe: Base de Estados

En esta fase se repiten las actividades “Capturar Señales”, “Remplazar Datos” y “Estimar Parámetros” ya descriptas en la fase inicial. A continuación se describen las actividades “Evaluar el Estado con M1”, “Decidir Estado”, “Emitir Alerta” e “Informar”. Las actividades “Evaluar el Estado con M1”, … , “Evaluar el Estado con Mn” tienen la misma estructura.

Una vez que completados estos pasos, se procede a mapear las actividades anteriormente descriptas a roles. Cabe destacar que este paso es casi trivial puesto que ya se cuenta con toda la información necesaria como para describir roles de acuerdo a la metodología GAIA. En este punto se agruparon actividades relacionadas para ser realizadas por un único rol.

En el atributo “Description” de cada rol, se describen las actividades asociadas. En el atributo “Protocols and activities” se enuncian las tareas que realizan dichas actividades y protocolos. Por último, en este caso completar los atributos “Permissions” y “Liveness properties” resulta casi directo debido a que las actividades ya especifican qué recursos leen o escriben y cuál es el orden de precedencia de cada una de sus tareas. A continuación se muestran los roles y protocolos obtenidos en este paso.

3.2 Modelo de roles

Role: Iniciador
Description: Realiza la actividad Iniciar Soporte.
Protocols and Activities: Comenzar actividad “Iniciar Soporte”
Permissions: reads supplied reads Base Modelos, KB Decision, Archivos Configuración. writes
Responsabilities Liveness: Iniciador = PedirCaracterísticasPaciente.VerificarCaracterísticas.GuardarCaracterísitcas.CargarParametrosReferencia

3.3 Modelo de interacción

Definición de protocolos asociados al rol Iniciador

IniciarActividadCapturarSeñales	
Iniciador	Caracterizador
Inicia la actividad	

IniciarActividadEstimarParámetros	
Iniciador	Caracterizador
Inicia la actividad	

IniciarActividadEvaluarEstado	
Iniciador	Referenciador
Inicia la actividad	

IniciarActividadDecidirEstado	
Iniciador	Alertador
Inicia la actividad	

Definición de protocolos asociados al rol Caracterizador

ActividarCapturarEstadoDeEstabilidad	
Caracterizador	Referenciador
Inicia la actividad	

ActividarEvaluarEstado	
Caracterizador	Evaluador
Actualiza la actividad	

Definición de protocolos asociados al rol Referenciador

InterrumpirDecisiónDeEstado	
Referenciador	Alertador
Interrumpe la actividad	

Definición de protocolos asociados al rol Evaluador

Informar_a_la_ActividadDecidirEstado	
Evaluador	Alertador
Genera datos de entrada a la actividad	

Definición de protocolos asociados al rol Alertador

ActivarActividadInformar	
Alertador	Informador
Reactiva su ejecución con sus parámetros	

3.4 Modelo de agentes

A continuación se muestra el mapeo entre roles y agentes.

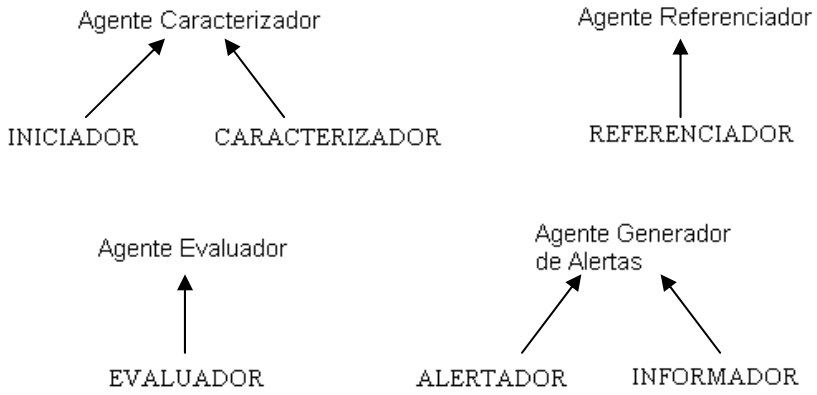


Figura 3. Mapeo de roles en agentes.

Finalmente se obtuvo la estructura final del sistema diseñado, que se muestra en la figura 4.

Dicho sistema resultó compuesto por los siguientes agentes:

- **Agente Caracterizador:** Es un agente reactivo dado que controla el comienzo y fin de los procesos, captura señales de los monitores, y compone las características a partir de las señales necesarias para el funcionamiento de los demás agentes.

- **Agente Referenciador:** Este agente analiza y construye un modelo de referencia a partir de los datos adquiridos durante el comienzo del monitoreo, asumiendo la hipótesis que durante el comienzo del monitoreo el profesional conduce al paciente a un estado estable. No incluye ningún modelo simbólico del mundo en el que se desenvuelve ni tampoco razonamiento simbólico, razón por la cual también es un agente reactivo.
- **Agente Evaluador:** tiene el rol de Evaluar el estado, según es aspecto que caracterice su modelo de comportamiento. Se podría decir que este agente es el encargado de verificar si realmente se está en una situación de emergencia.

- **Agente Generador De Alertas:** Se encarga de activar la alerta temprana requerida dependiendo del problema detectado.

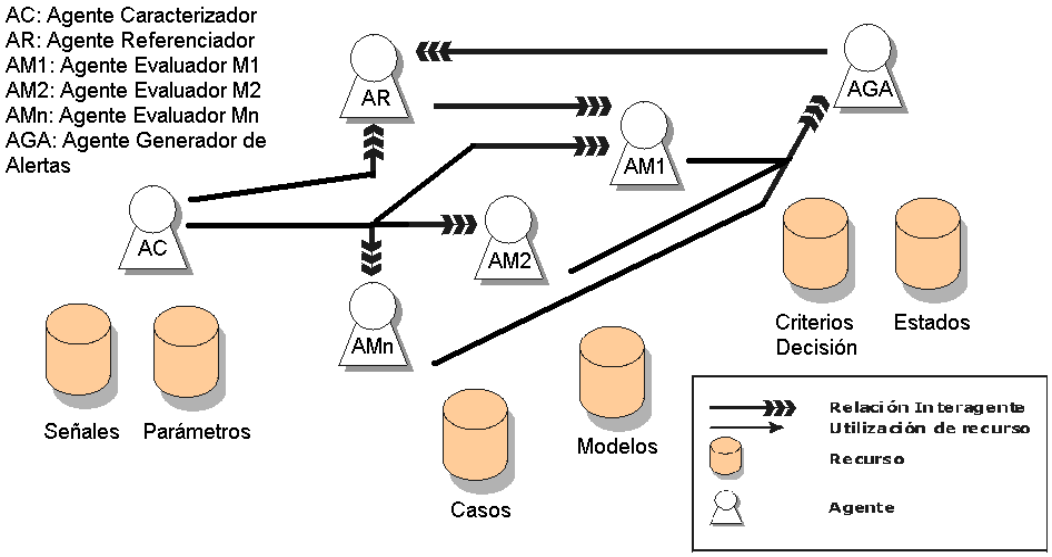


Figura 4. Arquitectura del SMA resultante aplicando Gaia.

Implementación en JADE

Para llevar el diseño del SMA detallado, a una aplicación en el entorno JADE, se deben hacer algunas suposiciones y definiciones. En primer lugar, para cada Rol de Gaia se consideró que la *liveness property* determinaba su comportamiento, de esta forma mediante los atributos de Jade: *Complex Behaviours* o *Simple Behaviours* es posible representar cada uno de los roles.

Por otro lado los roles junto al modelo de agentes sirven además para definir las estructuras de datos y módulos de software que serán utilizados por los agentes Jade. También se consideró que los métodos invocados por cada agente se corresponden con las actividades Gaia. Por último, a partir del modelo de interacción y protocolos de Gaia es posible definir los *ACL Messages* de Jade.

Teniendo en cuenta estas consideraciones para el mapeo entre las definiciones de Gaia y las de JADE, se pudo avanzar en la implementación del sistema.
En primer lugar se construyeron los agentes en Jade según lo establecido en la fase de diseño. Antes de avanzar en la asignación de los comportamientos, se decidió utilizar el modelo de interacciones para definir la estructura de mensajes a intercambiar entre los agentes, y así se definieron los ACL Messages y las secuencias de mensajes.

4.1 Definición de mensajes de la fase de inicio.

La primera actividad de la fase de inicio es “Iniciar Soporte”, la cual involucra a las tareas:

1. Iniciar actividad Capturar Señales.
2. Iniciar actividad Remplazar Datos.
3. Iniciar actividad Estimar Parámetros.
4. Iniciar actividad Evaluar el Estado con M1.
5. Iniciar actividad Evaluar el Estado con M2.
6. Iniciar actividad Evaluar el Estado con Mn.
7. Iniciar actividad Decidir Estado.
8. Iniciar actividad Emitir Alerta.

i. En el modelo de interacción, la definición de protocolos asociados al rol iniciador muestra que las tareas 1, 2 y 3, involucran al rol iniciador y al rol caracterizador. Si estos roles fueran llevados a cabo por diferentes agentes deberían enviarse mensajes para dar inicio a tales tareas, sin embargo, este no es el caso ya que como se vio en el modelo de agentes estos dos roles son llevados a cabo por el Agente Caracterizador, por lo que en la arquitectura de mensajes estas tareas corresponderán a métodos del mismo agente. Este agente es responsable de transmitir las señales fisiológicas capturadas desde los monitores a los demás agentes, empleando el estándar DICOM-waveforms.

ii. Para realizar las tareas 4, 5 y 6, se relacionan los roles iniciador y evaluador, por lo cual se deben enviar mensajes entre el agentes Caracterizador y los Evaluadores, quienes desempeñan tales roles.

iii. En la tarea 7 (8) se relaciona al rol iniciador con el referenciador (alertador) por lo que se deben intercambiar mensajes entre el Agente Caracterizador y el Agente Referenciador (Generador de Alertas).

De i, ii y iii, se establecen los mensajes necesarios para iniciar el soporte. En la figura 5 se esquematiza el envío de tales mensajes de iniciación.

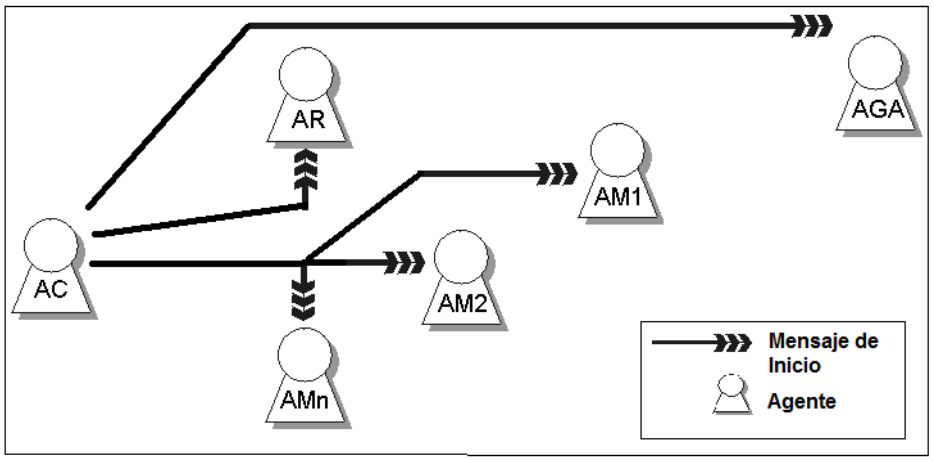


Figura 5. Envió de mensajes de iniciación.

A continuación se muestra la implementación del envío de estos mensajes en JADE.

```
//Envio de mensaje chequeando que los agentes iniciaron correctamente.
ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
msg.setContent(msj1);
msg.addReceiver(new AID(AR_AgentName, AID.ISLOCALNAME));
msg.addReceiver(new AID(AM1_AgentName, AID.ISLOCALNAME));
msg.addReceiver(new AID(AGA_AgentName, AID.ISLOCALNAME));
send(msg);
```

Además, cada agente que recibe un mensaje debe disponer de una rutina para la atención de los mismos. En JADE esto se realiza mediante un comportamiento cíclico (CyclicBehaviour). A continuación se muestra parte de la implementación de tal comportamiento.

```
addBehaviour(new CyclicBehaviour(this) { //para atender a los mensajes recibidos
public void action() {
ACLMessage  msg = myAgent.receive();
if(msg != null){
        if(msg.getPerformative()== ACLMessage.INFORM){//si recibe un mensaje INFORM
        content = msg.getContent();
        answerscont++;
        }
        else{
        System.out.println("[msj Unexpected] from "+msg.getSender().getLocalName());
        }

        }
        else {block();}//si no se reciben mensaje bloquear el comportamiento.
        }
});//fin CyclicBehaviour
```

Las próximas actividades en la fase de inicio son: “Capturar Señales”, “Remplazar datos” y “Estimar Parámetros”; estas actividades pertenecen a los roles iniciador y caracterizador que son llevados a cabo por el agente caracterizador, por lo que son métodos de dicho agente y no se deberán enviar mensajes inter-agentes. A continuación se muestra el método que invoca estas actividades, un CyclicBehaviour se encarga de temporizar la ejecución.

```
addBehaviour(new TickerBehaviour(this, 5000) {
protected void onTick(){
 //Espera q todos los agentes inicien.
 if(answerscont==3)
 {System.out.println("Todos los Agentes iniciaron correctamente");
 answerscont++; }
 if(answerscont==4)
 {
 // adquisición de señal desde monitor
 AdquiriendoSeñal();
 //estimación de parametros.
 EstimandoParametros();
 //Envio msj de aviso MRD y parametros.
 EnviandoParametros();
 }
 else {block();}
}
});//fin TickerBehaviour
```

La última actividad en la fase de inicio es “Capturar Estado de Estabilidad”, el rol que realiza dicha actividad es el referenciador, de esta forma el Agente Referenciador, según los parámetros que haya recibido desde el Agente Caracterizador, capturará la referencia de estabilidad.
Para llevar a cabo esta interacción se deberán enviar mensajes entre los Agentes Caracterizador y Referenciador, cuyo contenido sean los parámetros estimados en formato DICOM. En la figura 6 se muestra el envío de parámetros entre AC y AR.

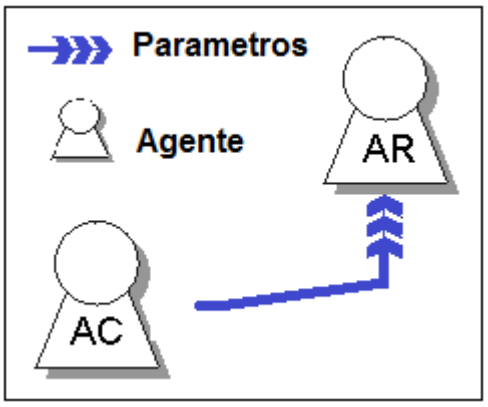


Figura 6. Envío de parámetros al Agente Referenciador.

De lo detallado hasta este momento, se debe tener en cuenta que el Agente Referenciador deberá poder discernir entre diferentes tipos de mensajes: aquellos cuyo contenido es una orden de activación y otros, cuyos contenidos serán los parámetros estimados de las señales de entrada.

A continuación se muestra porción del código para atender los mensajes que contienen parámetros.

```
if ("JavaSerialization".equals(msg.getLanguage())) {
    try {
        Parametros p = (Parametros)msg.getContentObject();
        System.out.println(getLocalName()+ "Se recibieron parametros");
        vector= p.valores();
    }
    catch(UnreadableException e3){
        System.err.println(getLocalName()+ " catched exception "+e3.getMessage());
    }

    CapturaEstabilidad();       //Captura estabilidad y segun sea normal o no
                                //envia mensaje de reporte o de Activacion y Alarma
}//if javaserialization
```

4.2 Definición de mensajes de la fase de soporte.

En la fase de soporte además de las actividades de la fase de inicio se realizan las actividades, “Evaluar el Estado con M1 ,…,Mn”, “Decidir Estado”, “Emitir Alerta” e “Informar”.

Para poder realizar la actividad “Evaluar estado” los agentes evaluadores AMn, necesitan recibir los parámetros desde el AC, y además una señal de aviso de que hay una posible inestabilidad informada por el AR. Dicha señal de aviso se generará en caso de que el AR una vez que ha capturado la referencia de estabilidad decide que hay cierta anomalía en los parámetros. Si no se detectan anomalías, la señal de aviso no se generará y el AR solo enviará al agente generador de alertas un informe de normalidad. En la figura 7 se muestra en azul el intercambio de parámetros entre AC y los evaluadores AMn, en verde el posible informe de normalidad y en rojo el posible aviso de inestabilidad.

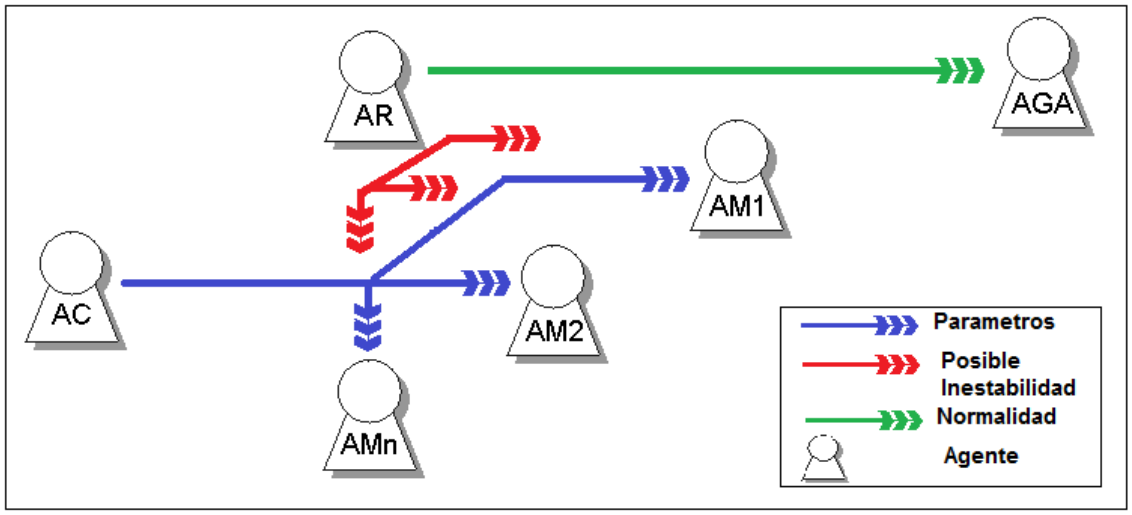


Figura 7. Distintos mensajes según el estado de estabilidad.

En caso de que se hayan activados los agentes evaluadores, realizaran la actividad "Decidir Estado" según su modelo de comportamiento. Los resultados de tal evaluación se informaran al agente AGA.

Seguidamente se muestra un ejemplo de la manera en que se desempeñan los agentes AMn.

```
protected void EjecutarModeloEvaluacion1(){

//Evalua estado según su modelo de comportamiento.
//Se verifica si realmente se está en una situación de emergencia.
//Segun resultado se envia reporte o alerta al AgenteAGA.

if(EstadoActivo==true){
    DeterminarEstabilidad();
    if(PEstable==true)
         {EnvioReporte();}
    else
         {EnvioAvisoAlerta();}
}
}//fin metodo
```

Una vez que el agente AGA recibe los informes de cada agente AMn, realizara las actividades "Informar" o "Alertar" según sea el caso. En la figura 8 se ve el envío de informe desde los AMn al AGA.

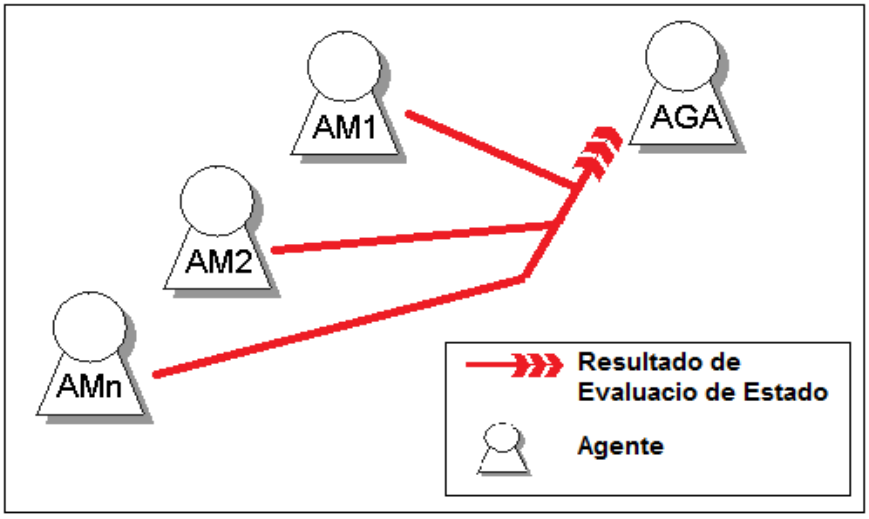


Figura 8. Envío de resultado de evaluación de estado..

A continuación se muestra un ejemplo del método de decisión de generación de alertas.

```
protected void DecidirGeneracionAlarma(){
//Se decide por votacion simple

if((AR_EE==false)&&(AM1_EE==false)){
System.out.println("ATENCION!!! EL PACIENTE PRESENTA ESTADO INESTABLE!!!");
}
else{if((AR_EE==true)&&(AM1_EE==true)){
     System.out.println("EL PACIENTE ESTA ESTABLE!!!");}
     else{System.out.println("NO SE PUEDE DECIDIR AUN!!!");}
}
}//fin del metodo
```

4.3 Funcionamiento del Sistema Multiagente

Para integrar lo presentado en las dos secciones anteriores, a continuación se describirá resumidamente la secuencia de funcionamiento del prototipo implementado bajo la arquitectura de JADE.

Desde la GUI de JADE se instancia al agente caracterizador (AC), que una vez que se activa se registra en el DF (Directory Facilitator) e instancia a los demás agentes: al agente referenciador (AR), al agente evaluador (AM1) y al agente generador de alertas (AGA). Cuando a su vez éstos últimos agentes se activan también se registran en el DF.
El DF ofrece un servicio de páginas amarillas por medio del cual un agente puede ofrecer servicios o encontrar a otros agentes que presten servicios que el necesite.

Posteriormente el Agente Caracterizador AC envía mensajes a los demás agentes, verificando que se hayan iniciado correctamente, y mediante un comportamiento cíclico atiende las respuestas.

Seguidamente ejecuta un comportamiento temporizado, que tras cierto retardo, adquiere señal de los monitores, si es necesario utiliza un modelo de reemplazo de datos faltantes (MRD), estima los parámetros y los envía a los agente evaluadores AM_N y al agente referenciador (AR).

El agente referenciador AR recibe desde AC el aviso de si se está utilizando el MRD y luego los parámetros. Este agente captura el estado de estabilidad y según sea normal o no, envía un

mensaje de “reporte normal” al Agente Generador de Alertas, o una señal de Activación a los Agentes Evaluadores a la vez que envía un “aviso de generación de alertas” al AGA.

En el caso de que AR haya detectado una no estabilidad y haya activado a los Agentes Evaluadores, éstos realizarán la evaluación de estado según su modelo de comportamiento y enviarán el respectivo “reporte normal” o “aviso de generación de alertas” según sea el caso.

Por último el Agente Generador de Alertas AGA, según los avisos y reportes recibidos, decidirá si emitir o no una determinada alerta.

4.4 Ejecución del SMA

En la figura 9 se muestra la instanciación del Agente Caracterizador bajo la interfaz gráfica de JADE.

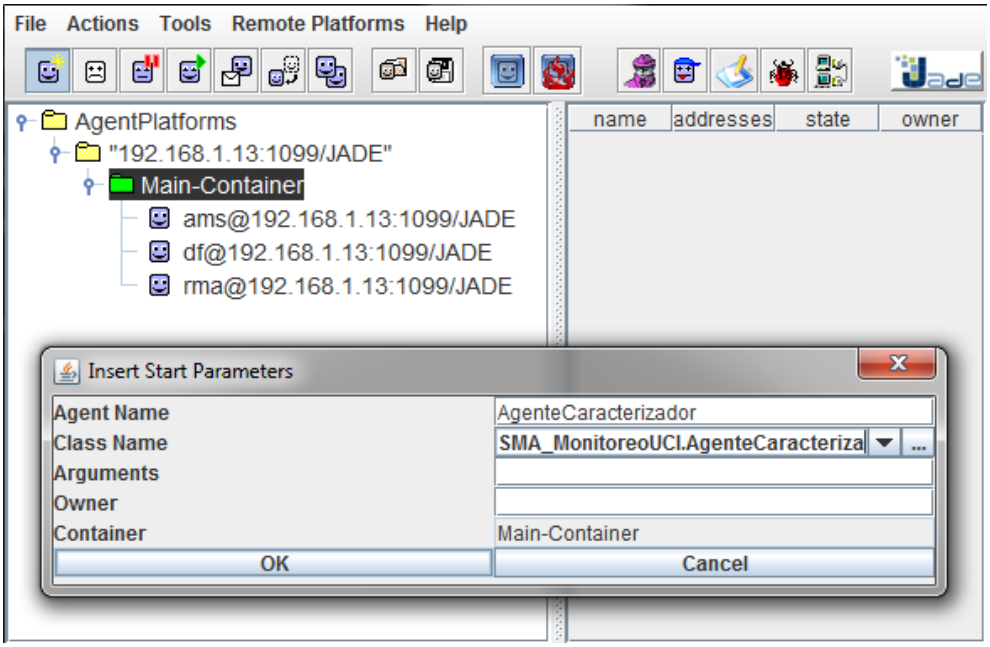


Figura 9. Instanciación del AC en la GUI de JADE.

Una vez activo el agente AC, su primer tarea es registrarse en el DF. A continuación se muestra la implementación de tal tarea.

```
//Registro en el DF
System.out.println(getLocalName()+" STARTED");

try {
        // create the agent descrption of itself
        DFAgentDescription dfd = new DFAgentDescription();
        dfd.setName(getAID());
        // register the description with the DF
        DFService.register(this, dfd);
        System.out.println(getLocalName()+" REGISTERED WITH THE DF");
} catch (FIPAException e) {
        e.printStackTrace();
}
```

Seguidamente se instancian los demás agentes, en la captura siguiente se muestra la instanciación del agente referenciador AR.

```
// create agent AR on the same container of the creator agent
try {
     AgentContainer container = (AgentContainer)getContainerController();
     AC_AR = container.createNewAgent(AR_AgentName,"SMA_MonitoreoUCI.AR", null);
     AC_AR.start();
     System.out.println(AR_AgentName+"CREATED ON"+container.getContainerName());
        }
catch (Exception any) {
     any.printStackTrace();
        }
```

La figura 10 es una captura del Agent Sniffer, se puede visualizar a todos los agentes inicializados y registrados en el DF, las flechas representan los mensajes que se intercambiaron.

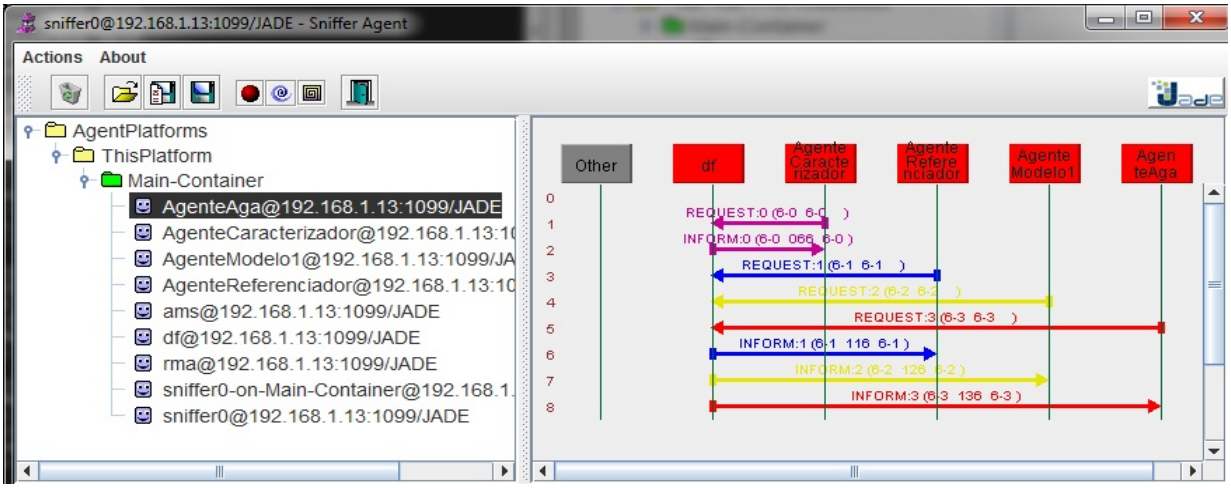


Figura 10. Intercambio de mensajes entre los agentes y el DF.

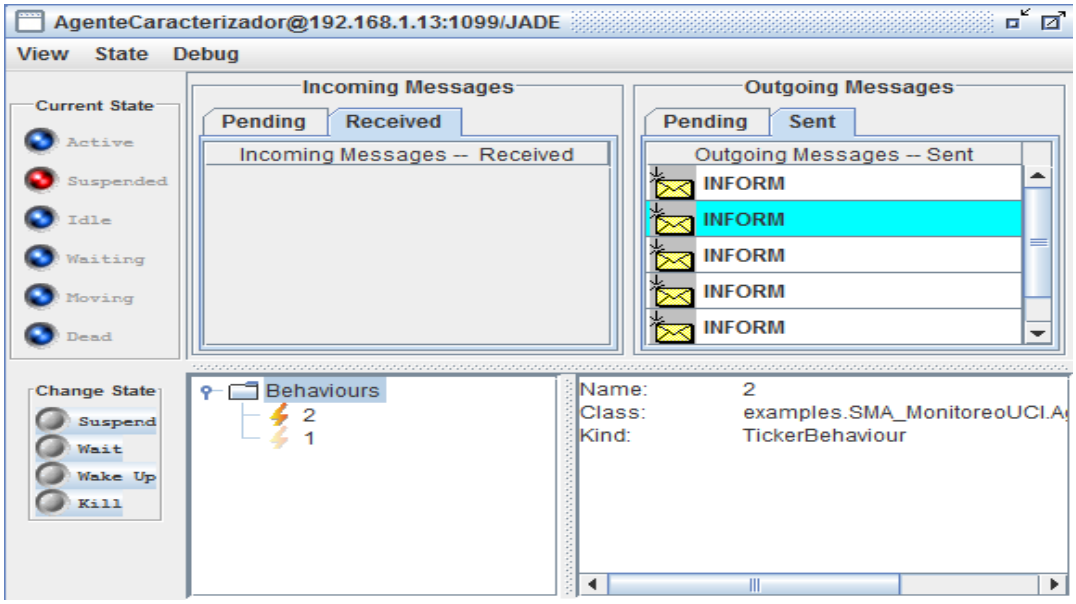


Figura 11. Agente Caracterizador enviado mensajes.

Las figuras 11 y 12 son capturas del Agent Introspector. La primera muestra al Agente Caracterizador enviando mensajes y la segunda muestra al Agente Referenciador recibiendo dichos mensajes.

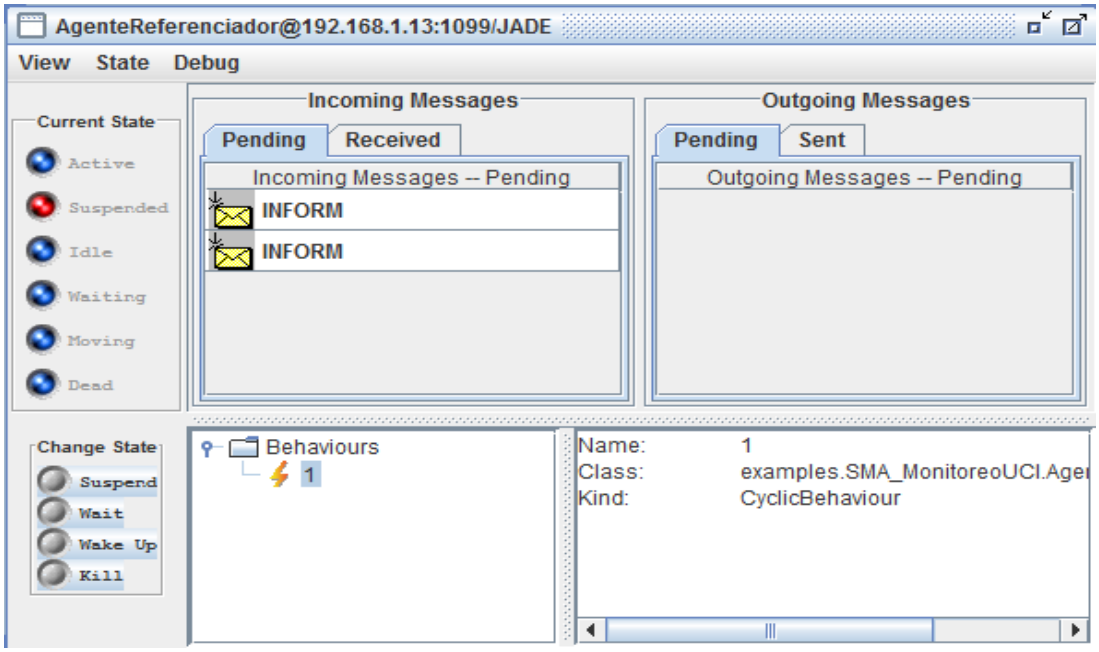


Figura 12. Agente Caracterizador enviado mensajes.

En la figura 13 se muestra la secuencia de mensajes en una condición de “Estado Estable” y en la figura 14 se observa la secuencia de mensajes en una posible condición de “Estado Inestable”.

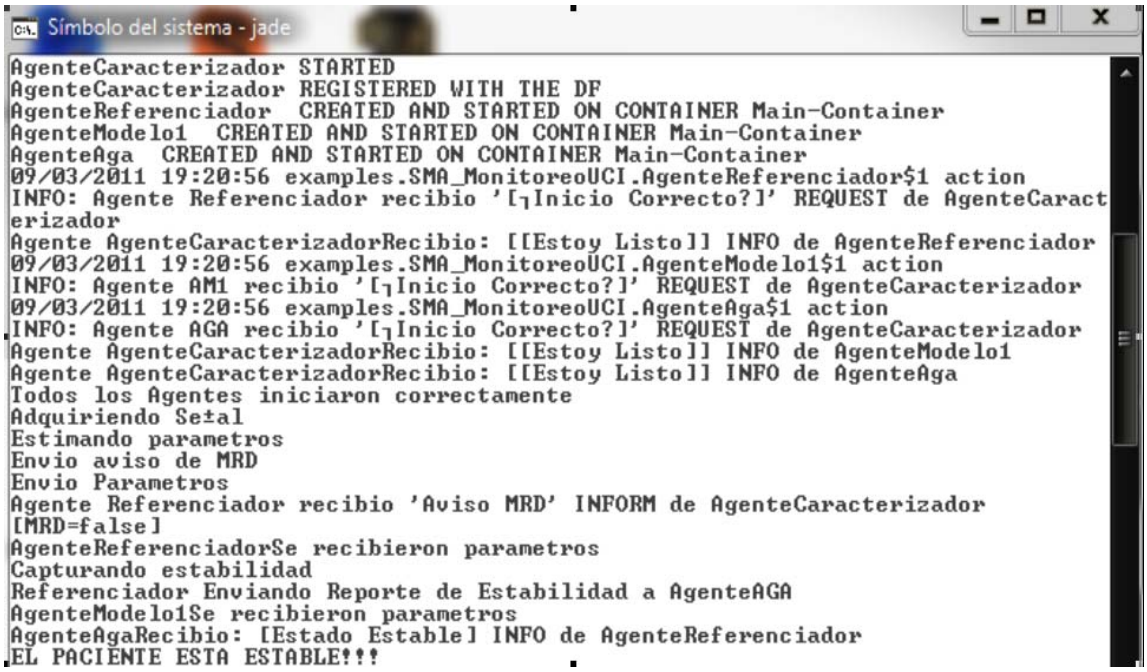


Figura 13.Secuencia de mensajes en un estado de estabilidad.

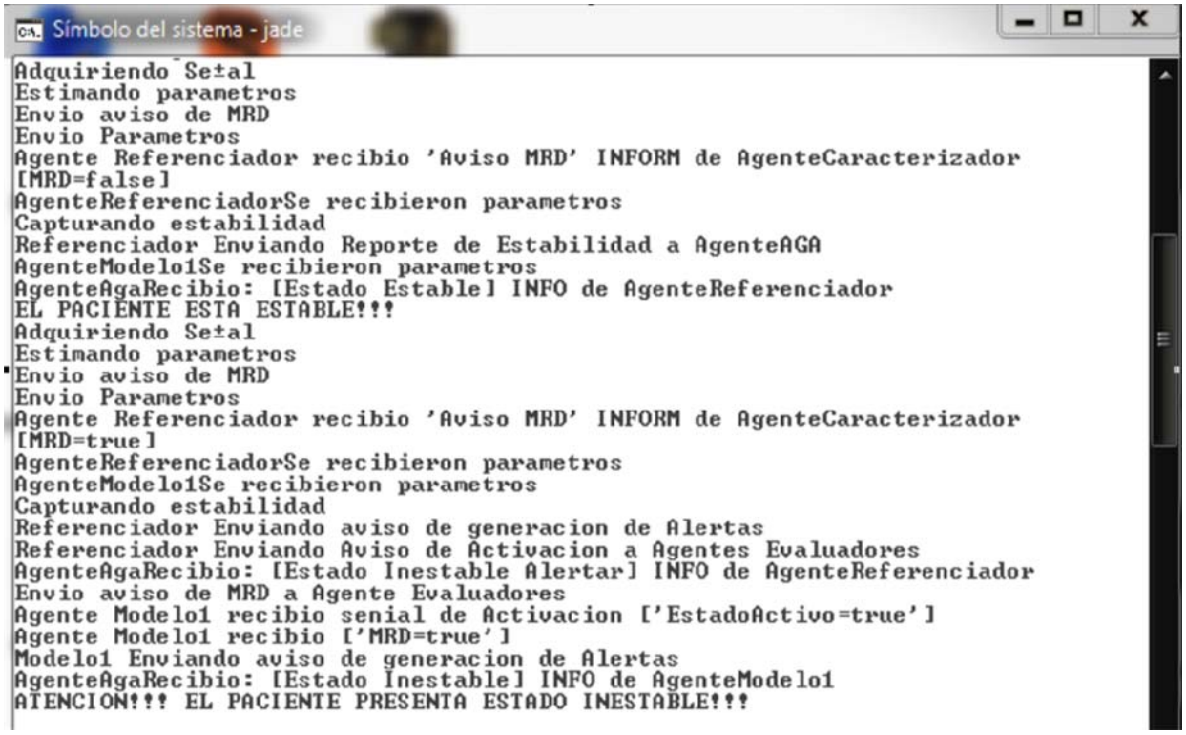


Figura 14. Secuencia de mensajes en un posible estado de inestabilidad.

Discusión

En este trabajo se trató el diseño de un sistema multiagente para el soporte en la toma de decisiones aplicado al ámbito de unidades de cuidado intensivo, haciendo principal hincapié en el tema de comunicación entre agentes.

Una de las características especiales que presenta este ambiente es la necesidad de trabajar con información de múltiples fuentes y diferentes formatos. En este sentido se propuso emplear el estándar DICOM Waveforms para modelar las señales y parámetros fisiológicos en el proceso de comunicación entre agentes.

Por otro lado la metodología GAIA, al igual que todas las metodologías para desarrollar MASs, se encuentran, hoy en día, en pleno desarrollo y discusión. Por ello, la descripción del uso de esta metodología en el análisis y diseño de un sistema del ámbito médico, puede significar un buen aporte para poder evaluar sus puntos débiles, fuertes y su aplicabilidad. En primer lugar, se comprobó la necesidad, planteada por Villareal et. al (2002), de incorporar a GAIA de una forma de descubrir los roles requeridos. Razón por la cual se aplicó esa aproximación con el objeto de formalizar el proceso de desarrollo.
El descubrimiento de los roles del sistema a través de la descripción de procesos y sus respectivas actividades para poder alcanzar la meta resultó una herramienta muy útil, permitiéndonos visualizar detalles que hubieran sido muy difícil de descubrir aplicando un modelo ad-hoc.

Referencias

[1] Wooldridge, Michael. An Introduction to Multiagent Systems. John Wiley & Sons, Baffins Lane, Chichester - England, 2002.

[2] Heine, C., Herrler, R., Petsch. M., Anhalt, C. (2003). ADAPT - Adaptive Multi Agent Process Planning & Coordination of Clinical Trials. In: AMCIS 2003 Proceedings, Tampa/Florida.
[3] Mohammed S., Fiaidhi J., Mohammed O. (2009). Sharing Biomedical Learning Knowledge for Social Ambient Intelligence. Journal of Computers, Vol. 4, No. 9, pp. 905-912. September 2009.
[4] Cheaib N., Otmane S., Djemal K., Mallem M. (2008). Groupware Design for Online Diagnosis Support. Image Processing Theory, Tools & Applications
[5] Wooldridge,M., Nicholas R. Jennings, David Kinny, The GAIA Methodology for Agent-Oriented Analysis and Design, J. of Autonomous Agents and Multi-Agents Systems. Vol 3(3), (2000).
[6] Villareal, P., M. Alesso, S. Rocco, M. R. Galli, O. Chiotti, Approaches for the Analysis and Design of Multi-Agent Systems, Revista Iberoamericana de Inteligencia Artificial. España (2002).
[7] B. Drozdowicz, A. Hadad, D. Evin, C. Böhm and O. Chiotti. Aspects Related to Methodological Design of Multi-agents Systems in Anesthesiology. Argentine Symposium on Artificial Intelligence (2003)