MASCONTROL: SISTEMA MULTIAGENTE PARA LA IDENTIFICACIÓN Y CONTROL DE SISTEMAS

E.J. González, A.F.Hamilton, J.A.Méndez, S.Torres, J.M.Torres, M. Sigut
Grupo de Computadoras y Control.
Av. Astrofísico Fco. Sánchez S/N. La Laguna, CP 38207. Tenerife
E-mail: ejgonzal@ull.es

Resumen

Se presenta un sistema multiagente (MAS) para la identificación y control de procesos. De un modo concreto, este MAS implementa el esquema de un regulador autoajustable (STR). En el diseño del MAS se ha adoptado las especificaciones FIPA, debido a que se han convertido en un estándar en el desarrollo de sistemas multiagentes y por otra parte no solamente implica especificaciones del lenguaje de comunicación entre los agentes, sino a especificaciones sobre la gestión de agentes, conversaciones... En este trabajo se ha incluido un Agente de Ontología (OA) que emplea DAML+OIL como lenguaje de implementación de la ontología. Por último se ha incluido en el sistema la herramienta Java denominada Evenet2000 para el diseño y entrenamiento de redes neuronales de arquitectura arbitraria. Dicha herramienta, a pesar de su objetivo inicial, se ha mostrado enormemente útil en su aplicación a problemas de control

Palabras Clave: Identificación, control de procesos, sistemas multiagente, ontologías

1 INTRODUCCIÓN

Los Sistemas Multiagente (MAS) [9] se han manifestado como una herramienta muy efectiva en un amplio rango de aplicaciones como es el caso de la Robótica o el comercio electrónico. Una de sus mayores aplicaciones se centra en la resolución de problemas distribuidos.

Sin embargo, las aplicaciones de la tecnología de agentes en la automatización de procesos no han sido tan numerosas. Las razones para ello se pueden resumir en las siguientes:

- Las necesidades de tiempo real de las aplicaciones de automatización de procesos son díficilmente alcanzables por la tecnología actual sobre agentes.

- Las características de las tareas de control de procesos, con interrelaciones entre las diversas variables controladas. Esto dificulta descomponer el problema de un modo adecuado para la aplicación de los agentes.

- La dificultad de encontrar paralelismo en el control de procesos. Este paralelismo podría ser modelado mediante el empleo de agentes.

Sin embargo, como contraposición, en este trabajo se presenta un MAS (denominado MASCONTROL [2,3]) para la identificación y control de procesos. Concretamente dicho MAS implementa el esquema de un regulador autoajustable (STR). Este problema puede ser tratado de forma adecuada mediante MAS debido principalmente por dos razones:

- Se pueden distinguir claramente varios módulos diferentes en un STR: identificación, estimación de los parámetros del controlador, adquisición de los datos...
- La mayor parte de dichos módulos trabajan de manera simultánea. De este modo se puede asignar un agente al comportamiento de cada módulo, optimizando de esta forma la capacidad de cálculo.

2 BREVE DESCRIPCIÓN DE LA PLATAFORMA DE AGENTES FIPA, EL LENGUAJE DAML+OIL Y LA HERRAMIENTA DE REDES NEURONALES EVENET2000

2.1 PLATAFORMA DE AGENTES FIPA. HERRAMIENTA FIPA-OS

Para el desarrollo de MASCONTROL, se ha optado por las especificaciones FIPA [1], debido principalmente a dos razones. Por un lado, dichas especificaciones se han convertido en un estándar reconocido en el desarrollo de sistemas multiagentes. Por otro lado, dichas especificaciones están referidos no únicamente al lenguaje de comunicación entre los agentes (como es el caso de otros estándares como el KQML) sino que abarcan aspectos como la gestión de los agentes en una plataforma o el flujo lógico en una conversación entre agentes. El modelo de

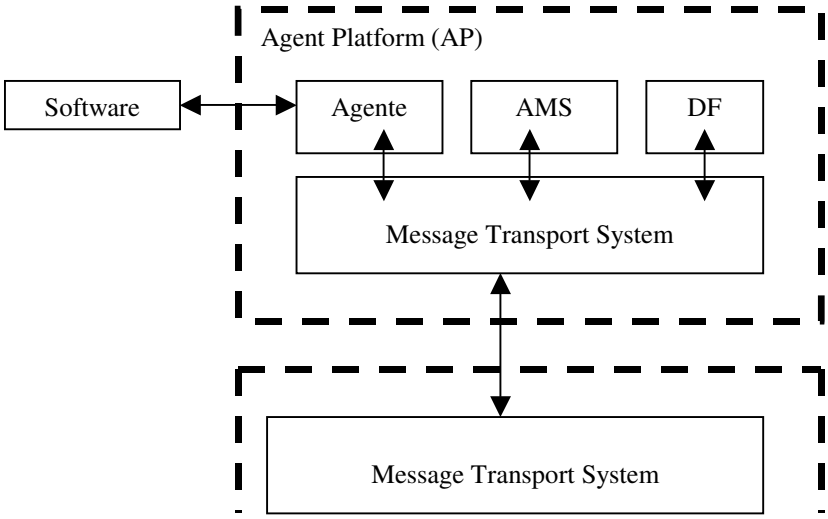

Figura 1: Modelo de Referencia de Agentes FIPA

referencia de agentes FIPA (*FIPA agent management reference model*, Figura 1) proporciona el marco en el cual los agentes existen y operan. El Facilitador de Directorios (*Directory Facilitator*, DF) proporciona un servicio de *páginas amarillas* a aquellos agentes que le consultan buscando los servicios suministrados por otros agentes de la plataforma. Por su parte el Sistema de Gestión de Agentes (*Agent Management System*, AMS) proporciona al resto de agentes un servicio del tipo *páginas blancas* y además mantiene un directorio que contiene las direcciones de los agentes registrados en la plataforma de agentes (*Agent Platform*, AP). Por último el Servicio de Transporte de Mensajes (*Message Transport Service*, MTS) constituye el método de comunicación por defecto entre los agentes de las diferentes plataformas. Una vez decidida la arquitectura de gestión del sistema, nos resta elegir la herramienta adecuada para su implementación. Las herramientas multiagente FIPA posibles son FIPA-OS, JADE, Zeus y Grasshopper (conviene aclarar que esta última, aunque diseñada inicialmente para OMG MASIF se ha adaptado para el estándar FIPA). Todas estas herramientas son multiplataforma, al encontrarse implementadas en el lenguaje de programación Java. Por tanto, cualquiera de ellas permite tratar el hecho de que los diferentes agentes accedan a la red interna a través de diferentes sistemas operativos. De las mencionadas herramientas, hemos seleccionado FIPA-OS [10]. Aparte de cumplir satisfactoriamente con las especificaciones FIPA, este marco de desarrollo presenta una estructuración de código por tareas. Esta estructura implica una depuración de código sencilla. Aparte de esta sencillez de codificación con FIPA-OS, la razón principal para su elección es la dedicación de un thread por tarea en vez de por agente, como en el resto de herramientas. Esto dota de una mayor robustez al sistema, puesto que evita que un posible bloqueo del *thread* inutilice completamente al agente.

2.2 ONTOLOGÍAS. EL LENGUAJE DAML+OIL

En este trabajo se ha incluido un Agente de Ontologías (*Ontology Agent*, OA). Un OA es un agente que posibilita el acceso a una o varias ontologías. Este es un punto clave, ya que disponer de una ontología común para la comunicación entre los agentes supone un formato común según el cual los agentes intercambian datos y conocimiento. Actualmente existen varios lenguajes de ontologías como el KIF o el OKBC. No obstante, se ha preferido optar por los conocidos como "lenguajes de marcas" (*markup languages*). Estos lenguajes ofrecen varias ventajas como la portabilidad de los datos, la facilidad de aprender y usar, flexibilidad... Además, la última generación de estos lenguajes proporcionan a los computadores un grado de autonomía extra que les puede ayudar a realizar su trabajo, siendo capaces de proporcionar toda clase de servicios adicionales más allá de los requerimientos estándar. La primera decisión a tomar sobre la inclusión de ontologías estriba en el lenguaje en que se van a representar éstas. Sería deseable que este lenguaje ofreciese la mayor riqueza semántica posible. En este sentido, el candidato más adecuado en la actualidad es el OWL. No obstante, el presente trabajo fue iniciado con anterioridad al lanzamiento de la especificación del OWL, por lo que se eligió como lenguaje de ontología el DAML+OIL [11]. A pesar de esto, la riqueza semántica de este lenguaje es suficiente para demostrar el poder de este tipo de lenguajes de marcas en su integración en sistemas multiagentes mediante el empleo de ontologías. En las secciones

posteriores se mostrarán varios ejemplos de definiciones en este lenguaje.

La decisión de tomar un lenguaje de ontologías basados en marcas, lleva implícita la elección de tres herramientas:

- Una herramienta de edición de la ontología de una forma gráfica (aunque el resultado de la edición es un fichero de texto). Se ha escogido en este caso la herramienta OilEd por dos motivos. En primer lugar, la capacidad de editar DAML+OIL viene en la distribución estándar (por tanto no es necesario incluir ningún *plug-in*). Además incluye la posibilidad de procesar directamente la ontología mediante un razonador FaCT.

- Un razonador que verifique la estructura de la ontología editada. En este punto, se han aplicado todos los razonadores analizados, esto es, FaCT (incluido en la distribución de OilEd), DAML+OIL Ontology Checker y DAMLValidator. De este modo se asegura, aunque de forma redundante en principio, de detectar completamente las posibles inconsistencias de la ontología.

- Un *parser* que permita extraer información de la ontología resultante. En este sentido es conveniente que esta herramienta estuviese implementada en Java, ya que de este modo la integración con la herramienta FIPA-OS (implementada en ese lenguaje) será más fácil. Con estas premisas, nos hemos decantado por Jena. Además, de este modo, prevemos que la transición a las nuevas versiones de esta herramienta, que incluyen soporte para el lenguaje OWL, sea asimismo no excesivamente compleja.

2.3 EVENET2000

Aparte de estas herramientas, se utilizan en el sistema las librerías de la herramienta Evenet2000 [4,5,6,7], desarrollada en el Grupo de Computadoras y Control de la Universidad de La Laguna. Esta librería fue diseñada inicialmente (y de forma independiente del desarrollo del MAS) para el entrenamiento de redes neuronales de arquitectura arbitraria. No obstante, esta herramienta se ha mostrado como muy versátil y útil en problemas de optimización en general, y relativos al control en particular [8]. La inclusión de esta herramienta en el MAS se justifica por tres razones principales:

- Su capacidad de optimización de funciones dependientes de unos parámetros es útil en la estimación de los coeficientes en la identificación del sistema a controlar.

- Al haber sido diseñada de forma independiente al estudio de los MAS, permite demostrar la capacidad y facilidad de inclusión en los MAS de herramientas ya existentes.

- Al ser una herramienta de código libre, permite realizar modificaciones para adaptarla a las necesidades del MAS. Debemos aclarar que no fue necesario realizar ningún tipo de modificación durante el desarrollo del MAS, lo que reafirma el punto anterior.

En este aspecto de inclusión de herramientas diseñadas dentro del Grupo CyC, una de las líneas abiertas consiste en la inclusión de los módulos de la librería de simulación y análisis de sistemas lineales de control, ControlWeb.

3 DISEÑO DE ONTOLOGÍA

En este trabajo se ha optado por una ontología de aplicación de grano grueso, puesto que el papel asumido por el OA no es elevado. Muchos de los conceptos y relaciones definidos no son empleados por el MAS. No obstante se han incluido en la ontología pensando en futuras ampliaciones del sistema.

En cuanto a las clases definidas, éstas hacen referencia, principalmente, a conceptos propios del control. En esta línea tenemos, por ejemplo, las clases *Planta*, *Sistema* (con las subclases *SistemaLineal* y *SistemaNoLineal*), *Comando*, *Error*... Otro conjunto de clases hacen referencia a los métodos de optimización de parámetros. Así tenemos *MetodoOptimizacion* (con sus subclases *MetodoOptimizacionLineal* y *MetodoOptimizacionNoLineal*), *Entrenamiento* (referido a los entrenamientos de optimización de los parámetros), *ConjuntoParametros* (el conjunto de parámetros a optimizar)...

Como ejemplo de definición de clases, el siguiente código DAML+OIL indica que la clase *SistemaNoLineal* es una subclase de *Sistema*.

```
<daml:Class rdf:about="#SistemaNoLineal">
  <rdfs:label>SistemaNoLineal</rdfs:label>
  <rdfs:comment>Sistema con ecuacion diferencial o
en diferencias no lineal</rdfs:comment>
  <rdfs:subClassOf>
    <daml:Class rdf:about="#Sistema"/>
  </rdfs:subClassOf>
</daml:Class>
```

Dentro de las propiedades definidas, el grueso de ellas se refieren a los valores de los conceptos definidos como clases: *valorPolo*, *valorCero*, *valorMetodoEntrenamiento*... El siguiente código,

por ejemplo, define que los objetos de la clase *Polo* tienen exactamente un valor de tipo *NumeroComplejo*

```
<daml:Class rdf:about="#Polo">
  <rdfs:label>Polo</rdfs:label>
    <rdfs:subClassOf>
      <daml:Restriction daml:cardinalityQ="1">
        <daml:onProperty rdf:resource="#valorPolo"/>
        <daml:hasClassQ
df:resource="#NumeroComplejo"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

Otra serie de propiedades se refieren a términos de la teoría de control como orden y tipo de un sistema. También se definen una serie de propiedades que facilitan la acción de MASCONTROL. En esta línea se han definido las siguientes propiedades

- *consignaAlcanzadaConControladorP*. Es una propiedad de dominio *Sistema* y rango *boolean*, que indica si la salida del sistema ha alcanzado la consigna mediante un controlador P.

- *accionControlAdecuada*. Esta propiedad tiene como dominio un *Sistema* y como rango una *AccionControl*. Indica la acción de control adecuada para que un sistema alcance la consigna deseada.

- *numeroVecesMejor*. El dominio de esta propiedad es *MetodoEntrenamiento* y de rango un *entero*. Indica el número de veces que el método de entrenamiento sujeto se ha mostrado como el mejor en una optimización.

Estas definiciones permiten al MAS realizar algunas inferencias interesantes a partir de los axiomas de la ontología. Por ejemplo, si un sistema no alcanza la consigna deseada con una acción de control P pura se puede concluir que se trata de un sistema de tipo cero. La acción de control adecuada para un sistema de tipo cero es la PI. Por tanto, un agente MASCONTROL puede deducir que la acción de control más adecuada para un sistema que no alcance la consigna con un controlador P es la PI, sin que esta última definición aparezca explícitamente. El código correspondiente es el siguiente:

```
<daml:DatatypeProperty rdf:about="#tipo">
  <rdfs:domain>
    <daml:Class rdf:about="#Sistema"/>
  </rdfs:domain>
  <rdfs:range>
    <xsd:integer/>
  <rdfs:range>
</daml:DatatypeProperty>
<rdf:Description rdf:about="#SistemaTipo0">
  <rdf:type>
    <daml:Class rdf:about="#Sistema"/>
  </rdf:type>
  <ns0:tipo>
    <xsd:integer xsd:value="0"/> </ns0:tipo>
  <ns0:vieneControladoPor
rdf:resource="#AccionPI"/>
  <ns0:consignaAlcanzadaConControladorP>
    <xsd:boolean xsd:value="false"/>
  </ns0:consignaAlcanzadaConControladorP>
</rdf:Description>
<rdf:Description
rdf:about="#SistemaTipo1oMayor">
  <rdf:type rdf:resource="#Sistema"/>
  <ns0:vieneControladoPor
rdf:resource="#AccionP"/>
  <ns0:consignaAlcanzadaConControladorP>
    <xsd:boolean xsd:value="true"/>
  </ns0:consignaAlcanzadaConControladorP>
</rdf:Description>
<daml:Class rdf:about="#Sistema">
  <daml:disjointUnionOf
rdf:parseType="daml:collection">
    <daml:Class rdf:about="#SistemaTipo1oMayor">
    <daml:Class rdf:about="#SistemaTipo0">
  </daml:disjointUnionOf>
</daml:Class>
```

A partir de estas definiciones, un *Sistema* puede ser un *SistemaTipo0* o un *SistemaTipo1oMayor* (no a la vez). Por tanto, un sistema cuya salida no alcanza la consigna cuando está controlado mediante una acción proporcional es unívocamente de tipo 0 (no existe otro sistema posible con esa característica). Por último, la acción de control adecuada es *AccionPI*.

Finalmente se declaran en la ontología una serie de instancias individuales. Entre estas declaraciones se encuentran las de los métodos de optimización disponibles en la herramienta Evenet2000: *GradienteDescendente*, *GradienteConjugado*, *AlgoritmoGenetico*, *EnfriamientoProgresivo* y *CaminoAleatorio*. Todos estos términos son instancias de la clase *MetodoOptimizacion*.

```
<rdf:Description rdf:about="#GradienteConjugado">
  <rdf:type>
    <daml:Class rdf:about="#MetodoOptimizacion"/>
  </rdf:type>
</rdf:Description>
```

Asimismo se declaran las instancias de las acciones de control implementadas (de la clase *AccionControl*) a saber, *AccionP*, *AccionPI* y *AccionRealimentacionEstado*. Las instancias de la ontología se complementan con unas instancias referidas a los polos en el origen (*PoloOrigen*, instancia de la clase *Polo*), y al tipo del sistema, distinguiendo entre los sistemas de tipo 0

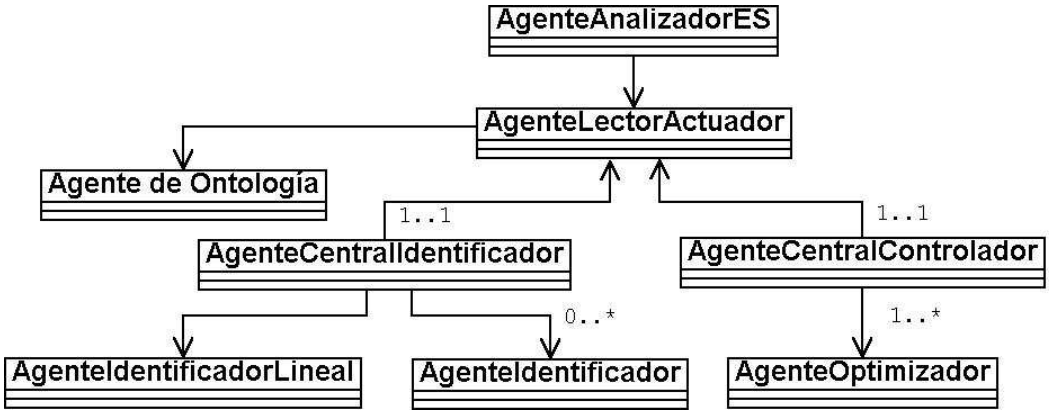

Figura 2: Arquitectura del sistema MASCONTROL

(*SistemaTipo0*) y el resto de sistemas (*SistemaTipo1oMayor*).

Conviene aclarar, tras definir la ontología a emplear, que la filosofía de ésta es totalmente distinta a la de otros sistemas como el COOL, puesto que éste únicamente describe los sistemas de control (interfaces, software, hardware). En nuestro caso se trata de una ontología empleable por los agentes del MAS para el funcionamiento del mismo.

4 ARQUITECTURA DEL SISTEMA

Poniendo en práctica el concepto de MAS, se ha dividido entre los agentes las diversas funcionalidades del sistema. La arquitectura MASCONTROL se compone de los siguientes tipos de agentes (Figura 2).

AgenteLectorActuador (LAA) Implementa el controlador indicado. Para ello muestrea la salida de la planta con un periodo fijado en su correspondiente fichero de perfiles XML. Dicho de otro modo, actúa como interfaz del MAS respecto a la planta. Almacena en su interior un vector con los pares de entrada-salida de la planta, lo que permitirá la identificación del sistema y determinar si la entrada es lo suficientemente rica para dicha identificación. También se encarga de mostrar las gráficas de evolución de la entrada y la salida. Este tipo de agente accede a un objeto de una clase que hereda una clase abstracta Java denominada *AccionControl*, encargado de calcular el comando que hay que suministrar a la planta en la siguiente etapa. Hasta el momento, las subclases implementadas se refieren a acciones de control proporcional, proporcional-integral y realimentación por variables de estado. Esto muestra la versatilidad de la herramienta, al implementar estructuras de control tan diversas. En cualquier momento, el LAA es capaz de modificar esta acción de control (cambiando el objeto que realiza el cálculo) basándose, por ejemplo, en estudios sobre el tipo del sistema.

AgenteIdentificador (IA) Cada uno de estos agentes se encarga de llevar a cabo la identificación del sistema a partir del conjunto de pares entrada-salida provenientes del LAA. Para ello hace uso de las librerías de optimización de la herramienta Evenet2000. El usuario, mediante el fichero de perfiles, o el CaIA, a partir del historial de MASCONTROL, escoge un método de optimización de los parámetros del sistema. En ambos casos, se han de suministrar los criterios de parada dependiendo del método escogido). El modelo del sistema es suministrado asímismo mediante el fichero de perfiles. Este modelo se encuentra en forma de red en un fichero generado en el editor gráfico del Evenet2000. Es decir, para un IA, el problema es equivalente al de optimización de los pesos de una red, cuyo patrón lo forman los pares de entrada/salida del sistema. Se ha incluido en el código del IA un predictor a un paso para la determinación del coste asignado.

AgenteIdentificadorLineal (ILA) Idéntico al anterior, pero supone un modelo que permita la expresión en forma de regresión lineal

AgenteCentralIdentificador (CIA) Sirve de gestor de los diversos agentes identificadores (ya sean lineales o no). Inicialmente consulta al DF sobre aquellos agentes registrados que prestan el servicio de identificación de un sistema. De forma periódica pide a estos agentes los resultados de la optimización que están llevando a cabo: valor de error, parámetros calculados y fichero donde se encuentra el modelo empleado. Con estos datos selecciona aquella optimización con unos mejores resultados y comunica al resto de agentes con el mismo modelo el conjunto de parámetros de mínima función de coste obtenido para dicho modelo. De este modo, los agentes de identificación partirán de este valor en la próxima optimización. En en cualquier momento,

este CIA puede solicitar a cada IA o ILA que cambie su modelo de sistema o cualquiera de los parámetros de la optimización. Para introducir un comportamiento más inteligente en el sistema, el CIA lleva una contabilidad de las veces que cada método de optimización ofrece el mejor resultado en la consulta periódica a los IAs. Esta información se almacena en un historial con el formato de DAML+OIL. De esta forma, aparte del uso de este fichero por parte del AgenteCargadorOptimización, en futuras versiones del MAS se puede añadir una mayor capacidad de inferencia a los agentes. En caso de pérdida de comunicación con alguno de los agentes identificadores, el sistema no se ve afectado, puesto que el CIA fija un tiempo máximo de espera de respuesta de los IAs y ILAs, de modo que si se ha superado ese plazo, el agente realiza los cálculos a partir de los datos recibidos. De esta forma se consigue que el sistema sea, en este sentido, tolerante a fallos.

AgenteOptimizador (OpA) Este agente realiza una optimización de los parámetros del controlador seleccionado. Para ello toma los valores procedentes de la identificación del sistema y los incluye como constantes en un sistema tratado como una red neuronal por los módulos de Evenet2000. Este modelo refleja el esquema de control deseado de tal modo que los parámetros del controlador son los pesos de la red. El modelo de controlador puede, a partir de la información proveniente del resto del MAS, ser cambiado de manera fácil en cualquier momento gracias a la modularidad de los elementos de Evenet2000.

AgenteCentralControlador (CCoA) Su cometido es similar al de CIA, esto es de gestión y supervisión, pero en este caso relativa a la optimización de los parámetros del controlador. Su labor se resume en los siguientes puntos:

1. De forma periódica solicita al CIA los detalles de la marcha del proceso de identificación: fichero donde se encuentra el modelo con mejores resultados en la identificación del sistema, pesos del sistema.

2. Tras analizar estos datos pide a cada OpA los valores de los parámetros del controlador que minimizan la función de coste, así como el valor de dicha función.

3. Entre los valores recibidos, determina el conjunto de parámetros de mínimo coste.

4. El CCoA almacena estos valores para su posterior envío al LAA.

5. Finalmente indica a todos los OpAs que detengan el entrenamiento actual y comiencen uno nuevo con los valores óptimos entre los de los entrenamientos anteriores con el modelo de planta pasado por el CIA. Para ello, los OpAs deben crear un fichero de texto que represente el esquema de control deseado y con el modelo de planta indicada por el CIA.

Del mismo modo que el CIA, este agente lleva una contabilidad de cuántas veces ofrece un mejor resultado cada OpA. Estos datos se almacenan en un fichero DAML+OIL, empleando términos definidos en la ontología diseñada. En cualquier momento, el CCoA puede solicitar a un OpA que modifique el modelo de su controlador.

AgenteAnalizadorES (AESA) Este agente opcional analiza los datos de entrada (comando enviado a la planta) y salida (lectura de la planta) proporcionado por el LAA. Este análisis se realiza en dos frentes. En primer lugar testea de forma aproximada si la entrada se ha estancado. Para ello determina el máximo y el mínimo en su valor en los últimos N pares (este valor es suministrado en un fichero de perfiles), y comprueba si esa cantidad es menor que un umbral (también fijado en el fichero de perfiles). En caso afirmativo, supone que la entrada no es lo suficientemente rica y sugiere al LAA que modifique el valor de la consigna, con lo que mejorará la identificación. Un proceso similar se realiza para determinar la estabilización de la salida. Esta información puede ser empleada por el MAS (mediante el OA) para estudiar el tipo del sistema (en caso de entrada escalón) o para sugerir un cambio de consigna para enriquecer la identificación del sistema. La opción de cambio de consigna puede deshabilitarse on-line

Agente de Ontología (OA) Se ha incluido en el sistema un agente de ontología, siendo este un punto novedoso en esta parte del trabajo y lo diferencia de sus antecedentes. En este caso el OA no interviene directamente en el control de la planta. La intervención de este agente se reduce a fecha de escritura de este trabajo a facilitar la información al LAA referente al tipo del sistema según su error en régimen permanente. Teniendo en cuenta la serie de definiciones incluidas en la ontología, son muchas las posibilidades de uso del OA en futuras implementaciones del MAS. En este sentido, por ejemplo, el CCoA puede consultar en un momento dado, las instancias de los métodos de optimización disponibles en el sistema. Con esa información puede crear un nuevo OpA que comience un entrenamiento empleando algunos de estos métodos. Además de estos agentes, se han diseñado una serie de agentes cuyo periodo de vida en el sistema es corto y se centran en la inicialización de MASCONTROL de un modo más sencillo.

AgenteCargadorOptimización (COA) Este agente es el encargado de leer el perfil indicado por el usuario en el fichero correspondiente e inicializa el CCoA y los OpAs (con sus parámetros de aprendizaje) en la misma JVM donde el COA es inicializado. Opcionalmente, y si así se indica por el usuario, localiza en el fichero de historial, en caso de existir, aquél método de optimización, incluidos el modelo y los parámetros del entrenamiento, que ha conducido a un mejor resultado para la planta concreta. La planta se indica mediante un identificador en el fichero de perfiles. Si no encuentra registrado un OpA con esos parámetros óptimos del historial, crea uno extra con esos criterios para el entrenamiento. De este modo, y como ese método concreto ha conducido a unos buenos resultados con anterioridad, es de esperar que el comportamiento del sistema mejore con la inclusión de este OpA extra.

AgenteCargadorIdentificador (CaIA) Realiza las operaciones equivalentes al COA pero respecto al CIA y los IAs. Por tanto, este agente es el encargado de leer el fichero de perfiles correspondiente, inicializando en la misma JVM estos tipos de agente. Del mismo modo, si así lo indica el usuario, localiza en el historial aquél método de identificación que ha conducido a un mejor resultado con anterioridad con la planta empleada. Si encuentra que no se ha creado un IA con esos parámetros crea uno de estos agentes extra con esos criterios de identificación.

La arquitectura propuesta permite optimizar la implementación de un sistema STR optimizado, ya que permite distribuir los cálculos en paralelo, en varias máquinas potentes, alejadas incluso de la planta objeto del control.

5 RESULTADOS

Las condiciones impuestas tanto por la velocidad en las comunicaciones en el interior de la red pero sobre todo por el tiempo de procesamiento del sistema nos obliga a imponer ciertas consideraciones sobre el tipo de planta a controlar. En concreto, se deduce de forma inmediata que el MAS propuesto no es adecuado para controlar sistemas excesivamente rápidos, o lo que es lo mismo, con una constante de tiempo demasiado pequeña, puesto que las diversas optimizaciones necesarias precisan un tiempo relativamente largo para alcanzar resultados fiables.

En este contexto, se ha escogido para las pruebas del MAS dos plantas existentes en los laboratorios gestionados por el grupo CyC de la ULL. Concretamente se ha empleado una planta de tanques interconectados y otra planta de control del nivel de agua en un depósito. Al comprobarse el funcionamiento del MAS con dos plantas distintas, se pretende que el modelo quede suficientemente validado. Por cuestiones de simplicidad se incluyen solamente los resultados referentes a la planta de tanques interconectados.

5.1 PLANTA DE TANQUES INTERCONECTADOS

5.1.1 Descripción de la planta

La primera planta con que se ha probado MASCONTROL ha consistido en una maqueta de tanques interconectados, concretamente el modelo CPI-100 (control de procesos industriales) de la empresa ALECOP. El esquema de esta planta (mostrado en la Figura 3) permite definir tres zonas bien diferentes:

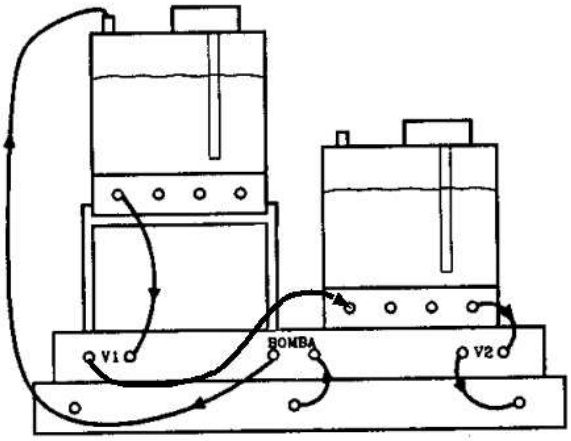

Figura 3: Esquema de la planta de tanques interconectados

- Depósito de trabajo. La maqueta está compuesta por dos depósitos (tanques) exactamente iguales, sobre los cuales se pueden realizar mediciones simultáneas y estudiar las interacciones existentes cuando se conectan según diversas configuraciones. La única diferencia entre ambos tanques es que uno de ellos está montado sobre una plataforma de elevación, para poder así variar la altura relativa de uno de los depósitos respecto al otro.

- Conexiones y control. Los depósitos van apoyados sobre una base de madera forrada. En su frontal aparecen tres grupos de dos tomas de entrada/salida, asociados a los elementos de control de flujo de líquido: 2 electroválvulas y una motobomba. Las electroválvulas son las encargadas de actuar de según el modo todo/nada para el paso del líquido de un tanque a otro.La maqueta está compuesta por dos depósitos (tanques) exactamente iguales, sobre los cuales se pueden realizar mediciones simultáneas y estudiar las interacciones existentes cuando se conectan según diversas configuraciones. La única diferencia entre ambos tanques es que uno de ellos está montado sobre una plataforma de elevación, para

poder así variar la altura relativa de uno de los depósitos respecto al otro.

- Depósito de drenaje: Mientras se está trabajando con la maqueta, se tiene una parte de líquido en los depósitos. El resto se encuentra en un depósito de drenaje que también almacena el total del líquido cuando no se trabaja con la maqueta. Un módulo electrónico convierte el valor del captor de nivel en un voltaje que es la magnitud concreta que se pretende controlar mediante el MAS. La lectura de esta magnitud se realiza mediante una tarjeta conversora analógica-digital de 8 bits con una velocidad de lectura de 32000 adquisiciones por segundo. Del mismo modo, la actuación sobre la planta se realiza a través de un conversor digital-analógico. El comando se va a encontrar limitado entre los valores de 0-5 V debido a la tarjeta que se aplica a la entrada de la etapa de potencia de la bomba. Por tanto, la entrada a la planta es el voltaje suministrado por la tarjeta que fija el flujo de líquido bombeado al tanque, mientras que la salida es la lectura en voltaje que representa la altura que alcanza el líquido en el segundo tanque.

5.1.2 Resultados

Con la arquitectura de agentes propuesta se realizaron varios experimentos con el objetivo de controlar la planta descrita. Los experimentos llevados a cabo se refieren a tres tipos de acciones de control: proporcional, proporcional-integral y realimentación por variables de estado. Conviene aclarar que la identificación se realizó, en todos los casos, a través de agentes simultáneos que optimizaban diversos modelos de la planta empleando diferentes métodos de optimización. Entre los tipos de modelos empleados destacamos los que implementan un sistema lineal. La Figura 4 muestra el esquema del modelo lineal de orden 2 construido mediante bloques de la herramienta Evenet2000. También destacaremos los que modifican este tipo de modelo incluyendo no linealidades. Un ejemplo de sistema no lineal se muestra en la Figura 5, el cual consiste en una modificación del modelo de la Figura 4, para incluir una no-linealidad mediante un bloque que representa una función tangente hiperbólica.

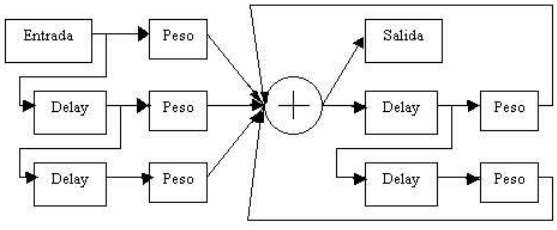

Figura 4: Esquema de bloques de un sistema de orden 2

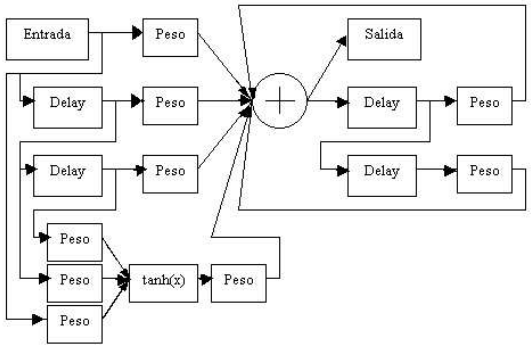

Figura 5: Esquema de un modelo de un sistema lineal de orden 2 modificado para incluir una no-linealidad.

Los resultados obtenidos se muestran a continuación.

Acción de Control P: La acción de control proporcional se implementa en dos puntos concretos de la arquitectura. En primer lugar en el objeto AccionControl unido al LAA. En este caso el objeto es de la subclase AccionP. El segundo sitio donde se indica es en el modelo de red cuyos parámetros se encarga de optimizar el conjunto de OpAs. En este caso el modelo es el de sistema de lazo cerrado con un único peso que multiplica al error. Respecto a este punto, la limitación en el valor del comando (entre 0 y 5 voltios) dificulta la optimización, introduciendo una no-derivabilidad en la función, modelada mediante un bloque cuya salida se encuentra acotada entre dichos límites. Debido a esta no-derivabilidad, no son adecuados los métodos basados en el gradiente. Por esta razón, en el proceso de entrenamiento se emplean sobre todo los algoritmos genéticos en los OpA. Puede definirse un enfoque alternativo definiendo una función de coste (derivable) que penalice enormemente los valores de comando fuera de los límites deseados.

Inicialmente se parte de un valor de $K_p=1$. Con este valor inicial se comienza una fase de aprendizaje donde continuamente se producen cambios de consigna en la planta, siempre evitando desbordamientos en los tanques e intentando dotar de suficiente riqueza a la entrada para mejorar la identificación. Se ha decidido que en esta etapa de aprendizaje la planta sea controlada por un PI, ya que se tiene la seguridad, a partir de la experiencia con este tipo de controladores, que es improbable que se produzcan efectos indeseados, por ejemplo, desbordamiento del sistema.

Cuando se estima que la identificación es conveniente, ya sea porque ha pasado suficiente tiempo, los parámetros se han estabilizado o se ha indicado de forma manual, se somete la entrada a la consigna deseada.

La gráfica de la Figura 6 muestra la optimización obtenida por los OpAs. Se intenta que el sistema siga los pares consigna, consigna (de modo que alcance la consigna pero penalizando los sobrepasamientosEsta penalización depende de la forma de la función de coste.). Para dotar de mayor información al patrón, éste se divide en cinco tramos con una secuencia creciente y decreciente. Los valores escogidos para la consigna son 3, 1, 2, 0.8 y 1.5. En esta figura se manifiesta que el sistema identificado es de tipo cero, puesto que el error en régimen permanente ante este esquema de realimentación no es nulo. Como es de esperar a partir de este dato, el sistema tiende a hacer crecer indefinidamente el valor de K_p , para disminuir lo más posible el coste producido.

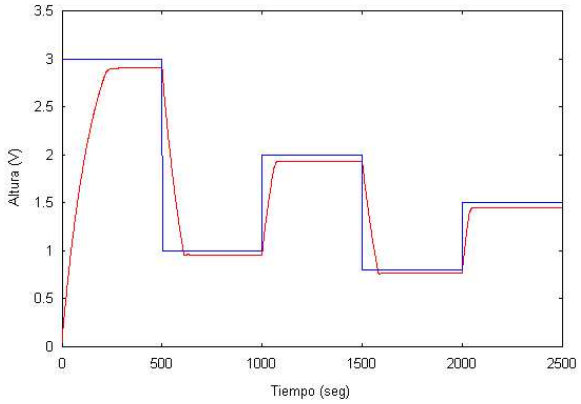

Figura 6: Optimización de los parámetros del controlador P por parte de los OpAs.

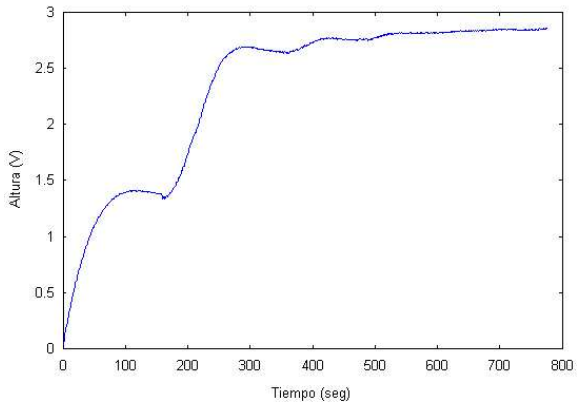

Figura 7: Control de la planta mediante una acción proporcional

La evolución de la salida de la planta ante una consigna de valor 3 voltios se muestra en la Figura 7. En esta figura se distinguen claramente las dos fases descritas: la de identificación (hasta un tiempo de 200 segundos) y la fase de consigna fija cambiando únicamente los parámetros del controlador. Como es de esperar, la salida de la planta no alcanza la consigna. Sin embargo, también se comprueba que a medida que aumenta el valor de K_p , disminuye el error en régimen permanente, tal como es de esperar y siempre dentro de las limitaciones de comando impuestas por el sistema.

Acción de Control PI: Tras la acción proporcional, se realizaron los mismos experimentos referidos a la acción proporcional-integral. Para ello se cambió el objeto de la clase AccionP por una instancia de la clase AccionPI y se modificó el modelo a optimizar por los OpAs. Estos son los dos únicos cambios que es necesario incluir en el sistema, debido a la modularidad con que está construída la arquitectura.

Si analizamos de nuevo la optimización realizada por los OpAs (en esta ocasión los valores de las consignas son distintos a los anteriores), nos encontramos con el comportamiento mostrado en la gráfica de la Figura 8. Como era de esperar, la salida del sistema optimizado alcanza el valor de la consigna en cada tramo, presentando un sobrepaso pequeño.

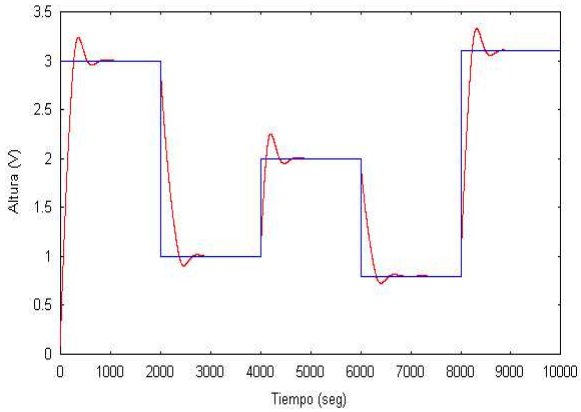

Figura 8: Optimización de los parámetros del controlador PI por parte de los OpAs.

Esta mejora se refleja en la salida real de la planta (ver Figura 9). Como se comprueba, ésta alcanza el valor de consigna (en este caso 3), además de que el sobrepaso es pequeño.

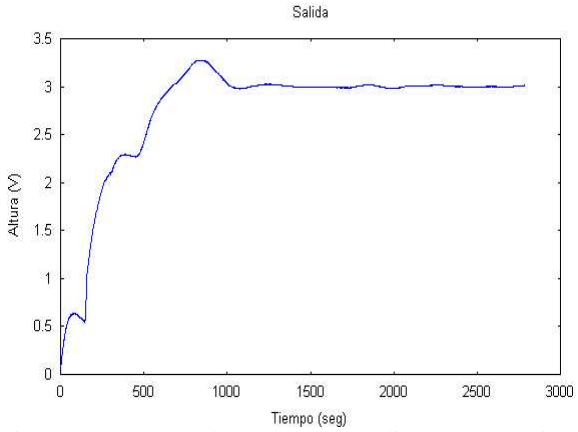

Figure 9: Control de la planta mediante una acción proporcional-integral.

El proceso, como en el caso de la acción proporcional, se ha realizado en dos fases, una de entrenamiento y otra de consigna fija.

Tal como se ha comentado anteriormente, el sistema de agentes crea de forma periódica una serie de ficheros (en DAML+OIL) con información de cuántas veces ha sido un entrenamiento y un modelo los mejores a la hora de optimizar los parámetros del sistema. El siguiente código es un extracto de uno de estos ficheros:

```
<RDFNsId2:Entrenamiento rdf:about="entr1"
  RDFNsId2:numeroVecesMejor="47">
<RDFNsId2:ValorRed>redes/modelosistematanques
2.red
   </RDFNsId2:ValorRed>
</RDFNsId2:Entrenamiento>
```

En un futura implementación de la herramienta Evenet2000, se pretende que los ficheros de red sean estructurados en el lenguaje de marcas XML. Este hecho podría traducirse en que la información de la red pueda almacenarse directamente en DAML+OIL en vez de un fichero independiente.

El caso de la acción proporcional-integral nos ilustra sobre lo adecuado de emplear un agente de ontología en un MAS destinado a un problema tan aparentemente alejado del empleo de esta herramienta. En la Figura 10, se muestra la evolución de la salida de la planta inicialmente con una acción proporcional pura. En un momento dado, entorno a los 650 segundos, el sistema multiagente (mediante el agente AESA) se da cuenta de que la salida se ha estabilizado en un valor alejado de la consigna. Entonces, el LAA pregunta al OA sobre qué acción de control le va bien a una planta cuando su salida no alcanza la consigna mediante una acción proporcional. El OA busca en su ontología y le responde que la acción buscada es una acción proporcional-integral. El sistema entonces realiza las modificaciones necesarias (acción de control y modelo de la red) para cambiar a una acción proporcional-integral. De nuevo resaltamos que este es un punto novedoso en el tratamiento de los problemas de control mediante sistemas multiagente. La salida de la planta, como era de esperar, alcanza, ahora sí, el valor de la consigna (3 voltios).

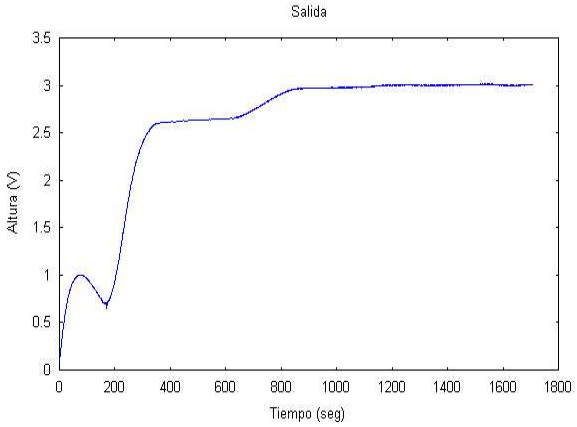

Figura 10: Control de la planta mediante una acción proporcional modificada a una acción proporcional-integral.

Realimentación por Variables de Estado: Habiendo comprobado la bondad de la arquitectura de agentes MASCONTROL para entrenamientos del tipo proporcional y proporcional-integral, el próximo punto consiste en demostrar lo adecuado de la arquitectura para un sistema de control basado en otra acción completamente diferente: la realimentación por variables de estado. Como se deduce a partir del modelo de control, el controlador es más dependiente de la bondad de la identificación. En este sentido, por ejemplo, para un controlador PI, el comportamiento de la salida lleva a alcanzar la consigna independientemente de que el sistema haya sido identificado convenientemente o no. En el caso de la realimentación por variables de estado, al menos la ganancia de la planta debe ser determinada adecuadamente para alcanzar la consigna.

A nivel práctico, se deben realizar algunas modificaciones menores en el sistema. La primera de ellas, claro está, se debe producir en el objeto AccionControl, que ahora debe implementar una realimentación por variables de estado. En concreto, suministra el comando sobre la planta a partir de la consigna y los valores de entrada y salida anteriores (se emplea la forma canónica controlable para la definición de las variables de estado). La otra modificación estriba en que ya carece de sentido la optimización de los parámetros del controlador. Simplemente se calcula ante una nueva identificación el valor de los parámetros K y de Q. Desde un punto de vista práctico, esto provoca que, en este caso, el CCoA se limite a calcular estos valores.

Los resultados obtenidos se muestran en las gráficas agrupadas en la Figura 11. En este caso, la fase de identificación se toma más larga que la realizada en los controladores P y PI, en búsqueda de que la identificación sea más efectiva. Por tanto, la salida de la planta en fase de identificación presenta picos, correspondientes a los tramos de diferente consigna.

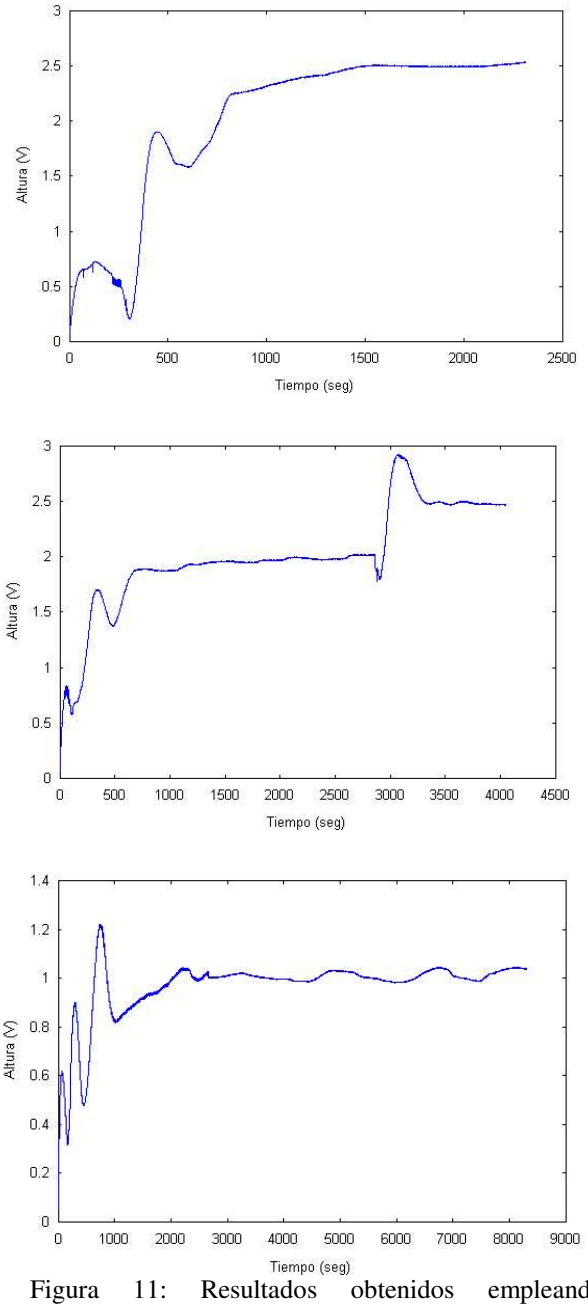

Figura 11: Resultados obtenidos empleando realimentación por variables de estado

Referencias

[1] García Alonso D., Pavón Mestras J. "Introducción al estándar FIPA". UCM-DSIP 98-00. Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid. 2000.

[2] González E.J. "Diseño e Implementación de una Arquitectura Multipropósito basada en agentes inteligentes: Aplicación a la planificación automática de agendas y al control de procesos". Tesis Doctoral. Universidad de La Laguna 2004.

[3] González E.J., Hamilton A.F., Moreno L. Marichal R.L, Muñoz V., "MASCONTROL: A MAS for System Identification and Process Control", Aceptado en el 8th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2004.

[4] González E.J., "Diseño e Implementación en Java de una Librería y un Entorno de Cálculo para Entrenamiento de Redes Neuronales Basados en la Técnica de Grafo de Flujo de Señal". Memoria de Licenciatura en la Universidad de La Laguna, 2000

[5] González E.J., Moreno L., Hamilton A., Piñeiro J.D., Marichal R.L., Marichal G.N. "Evenet2000: A New Java-Based Neural Network Toolkit". Proceedings of the Second ICSC Symposium on Engineering of Intelligent Systems (EIS 2000), Paisley, Escocia. Junio de 2000.

[6] González E.J., Moreno L., Hamilton A., Sigut J., Marichal R.L. "Evenet-2000: Designing and Training Arbitrary Neural Networks in Java". Bio-Inspired Applications of Conectionism. LNAI Springer Verlag. 2001.

[7] González E.J., Moreno L., Hamilton A., Aguilar R.M., Marichal R.L. "Neural Networks Teaching using Evenet-2000". Computer Applications in Engineering Education. 11(1). 1-5. 2003.

[8] González E.J., Hamilton A., Moreno L., Marichal R.L. "Evenet2000 as Software Application for System Identification and Control". Proceedings of the 11th Mediterranean Conference on Control and Automation MED03. Rodas, Grecia. Junio 2003.

[9] Honavar V. "Intelligent Agents and Multi-Agent Systems". IEEE Conference on Evolutionary Computation (CEC), Washington, 1999.

[10] Laukkanen M., "Evaluation of FIPA-OS 1.03", Cellular System Development, Sonera Mobile Operator, Sonera Ltd. Technical Report. 2000.

[11] Zou Y., Finin T. Peng T. Joshi A., Cost S. "Agent Communication In DAML World", First GSFC/JPL Workshop on Radical Agent Concepts, 2001.