# Plataforma para Implantar Sistemas de Supervisión y Control Basados en Agentes Inteligentes ⋆

**Leandro León** * **Addison Ríos-Bolívar** * **Francisco Hidrobo** *
**Jose Aguilar** *

* *Universidad de Los Andes*
*Facultad de Ingeniería*
*CEMISID*
*Mérida 5101, Venezuela*
*e-mail:* rleon,ilich,hidrobo,aguilar@ula.ve.

Resumen: En esta contribución se presenta una plataforma que permite la implantación de sistemas de control y supervisión basados en agentes inteligentes. Esta plataforma constituye un Medio de Gestión de Servicios (MGS) para sistemas multiagentes (SMA); en particular, ofrece servicios de comunicación y gestión para agentes en entornos con restricciones de tiempo real. Con esta plataforma se pueden concebir aplicaciones como sistemas multiagentes especializados, definidos para coordinar, ejecutar y evaluar tareas de control y supervisión necesarias en el procesamiento de la información del proceso y la toma de decisiones en procesos técnicos. Los agentes de control y supervisión interactúan con un sistema multiagente que modelan los elementos de las unidades de producción a través de abstracciones lógicas y funcionales de los procesos. Las complejas interacciones, propias de los sistemas de automatización, son soportadas por el MGS que provee diferentes mecanismos de comunicación, en base a las características funcionales de los agentes.

Palabras Claves: Medio de Gestión de Servicios. Agentes Inteligentes. Control Automático. Sistemas a Tiempo Real. Automatización.

## 1. INTRODUCCIÓN

Los sistemas de automatización industrial son aplicaciones que se caracterizan por requerimientos bien específicos de productividad y de seguridad operacional. Esto es, la automatización industrial debe satisfacer requerimientos de seguridad, confiabilidad, eficiencia y calidad [Aguilar et al., 2004]. Por otro lado, para la automatización de procesos se usan sistemas de software y hardware de gran escala, complejos, distribuidos y persistentes, los cuales son definidos en función de las características de los procesos técnicos a ser controlados y supervisados [Pinto, 2000]. El rápido desarrollo de componentes de hardware de alta capacidad y de tecnologías de información y comunicación (TIC) liderizan una fuerte necesidad de integración en sistemas automáticos, donde se requieren tareas de supervisión y control altamente complejas. Esto ha obligado a pensar en nuevos paradigmas basados en la teoría de inteligencia artificial distribuida para el diseño de herramientas que implementen control y supervisión inteligente. De esa teoría, el paradigma de desarrollo de software orientado a agentes permite diseñar sistemas sofisticados y complejos. Un Agente de Software es una entidad de software que encapsula datos y códigos, y se puede ejecutar dentro de su propio hilo (thread) de control. La decisión de cómo y cuándo realizar una acción es controlada por el mismo agente. Esto es, el agente tiene la capacidad de ejecutar una acción de manera autónoma sin ser invocado externamente. Puede ser visto como un objeto proactivo, Estas propiedades divergen de las entidades de software pasivas (como son conocidos clásicamente a los objetos), en las cuales se requiere de una interacción remota [Albert et al., 2003]. Las propiedades más importantes de los agentes son: autonomía, comunicación, sociabilidad, capacidad de reacción, inteligencia y movilidad. Estas propiedades permiten que la tecnología de agentes pueda ser utilizada para satisfacer requerimientos para la automatización de procesos, tales como la supervisión y el control inteligente [Weiss, 1999, Aguilar et al., 2007].

Típicamente, las tareas de automatización de procesos no han sido desarrolladas como una aplicación de las nuevas tecnologías de información, entre las que se encuentran los agentes inteligentes. Sin embargo, algunas investigaciones se han orientado hacía el uso de la tecnología de agentes en la implementación de sistemas de automatización de procesos [Bratoukhine et al., 2002, Jennings and Bussmann, 2003, Pakonen, 2004, Wagner, 2003, Hidrobo et al., 2005]. En general, las aplicaciones se han caracterizado por la realización de un acoplamiento entre los principios operacionales de los sistemas de automatización de procesos y los agentes inteligentes, permitiendo obtener sistemas distribuidos y de ingeniería complejos. Los sistemas automatizados se pueden representar mediante diferentes niveles, cada uno de los cuales tiene características

⋆ Trabajo financiado por el FONACIT bajo el proyecto **2005000170** y por el CDCHT-ULA **I-1103-08-02-A**.

operacionales adecuadas: nivel de dispositivos de campo (nivel operacional), para la captura de la información de los procesos, nivel de control supervisorio y optimización (nivel táctico), donde se ejecutan las tareas de control, y nivel de gerencia de los procesos (nivel estratégico), donde se evalúan y desarrollan las estrategias de producción. Esta arquitectura de operación jerárquica permite la distribución de las funcionalidades de las actividades de automatización a través de la descripción de las diferentes tareas operacionales, tácticas y estratégicas. Fundamentalmente y de manera tradicional, la inteligencia reside en los niveles superiores [Ríos-Bolívar et al., 2007].

En este mismo sentido, el paradigma de agentes inteligentes es una manera natural de descomposición de sistemas, y una alternativa razonable para implantar las funcionalidades de automatización en los diferentes niveles (Bravo, 2005; Wagner, 2002). Así, los niveles de un sistema automatizado se pueden representar por sub-sistemas y componentes de los sub-sistemas, los cuales son definidos por agentes y comunidades de agentes. Tradicionalmente, las aplicaciones de agentes en control y supervisión de procesos no se diseñan para que satisfagan esas características, por el contrario, se parte de que los controladores son sistemas reactivos, y a partir de allí se definen los agentes para las tareas de supervisión de esos controladores [Seilonen et al., 2003, Bravo et al., 2005]. Esto representa una primera alternativa de aplicación de Agentes Inteligentes en la automatización industrial, tal como se muestra en la Figura 1.

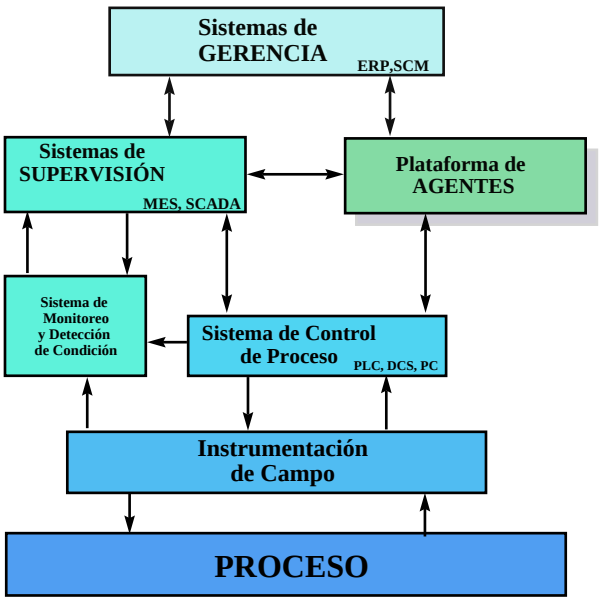


Figura 1. Agentes integrados en automatización industrial.

En esta manera de aplicación de agentes persisten todos los sistemas de control en tiempo real que se conocen hoy día (PLCs, SCADA, DCS, etc.), y los mecanismos de monitoreo y supervisión que soportan el diagnóstico del proceso productivo. Los agentes pueden ser aplicados para optimizar la producción. Otra alternativa, consiste en aplicar la filosofía de agentes inteligentes en todos los niveles de la automatización, tal como se muestra en la Figura 2.

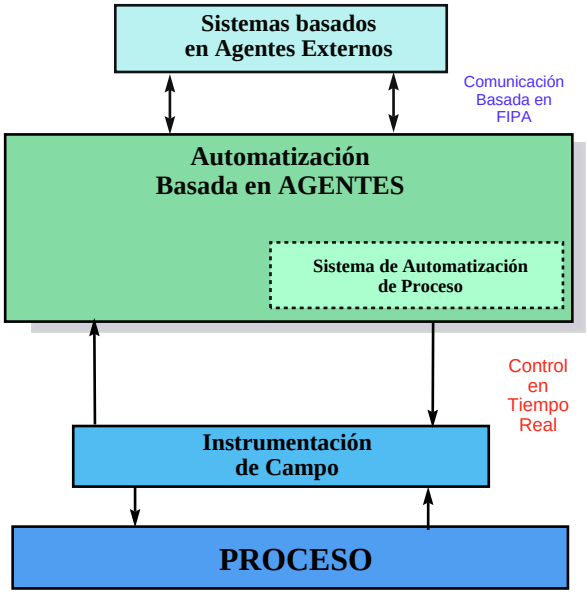


Figura 2. Agentes en automatización integrada.

Las interacciones entre sub-sistemas y componentes son definidas por mecanismos de cooperación, coordinación y negociación. Por medio de mecanismos explícitos se establecen las relaciones entre sub-sistemas y componentes. Por lo tanto, la inteligencia puede ser distribuida en los distintos niveles. De esta manera, el paradigma de agentes inteligentes es adecuado para cumplir con los requerimientos de los sistemas de automatización actuales, donde la reconfigurabilidad y la flexibilidad, conjuntamente con la inteligencia, son aspectos importantes a satisfacer [Jennings and Bussmann, 2003]; [Seilonen et al., 2003, Wagner, 2003].

El uso de la tecnología de agentes para el diseño de sistemas de automatización de procesos facilita la captura de las propiedades inherentes en estos sistemas. Sin embargo, la implantación de los servicios requeridos para soportar los elementos de diseño en SMA puede ser una tarea muy tediosa para usuarios no especializados en desarrollo de software, con restricciones de tiempo real. En este sentido, existen algunas propuestas que presentan la solución a través de un Medio de Gestión de Servicios; entre éstas destaca JADE [1], la cual es una implantación del estándar FIPA [2], [Bellifemine et al., 2003]. La principal desventaja de JADE es que no toma en cuenta las restricciones tiempo real para el manejo agentes.

En este trabajo se formula una arquitectura que permite el desarrollo de Sistemas Multiagentes (SMA) para el control y monitoreo de procesos industriales, que permiten la operación segura y óptima en base a los objetivos de producción. El esquema de desarrollo se fundamenta, primeramente, en la definición de un Medio de Gestión de Servicios basado en FIPA, y en segundo lugar, en la definición de la arquitectura de implantación del SMA en un ambiente de automatización industrial, en conjunción con la especificación de los agentes [Aguilar et al., 2007, OMG, 2003]. En el marco de este trabajo, y con el objetivo

[1] Java Agent DEvelopment Framework
[2] Foundation for Intelligent Physical Agents, *www.fipa.org*

de fijar las bases que conllevan a la implantación de los agentes, se presenta todo la arquitectura que debe soportar aplicaciones de control y supervisión de procesos con la filosofía de Agentes Inteligentes.

## 2. PROPUESTA DE AUTOMATIZACIÓN INDUSTRIAL BASADO EN AGENTES

### *2.1 Modelo de referencia*

Un modelo de referencia para automatización integrada de procesos de producción ha sido reportado en [Hidrobo et al., 2005], cuyos elementos de base se describen, brevemente, a partir de la Figura 3:

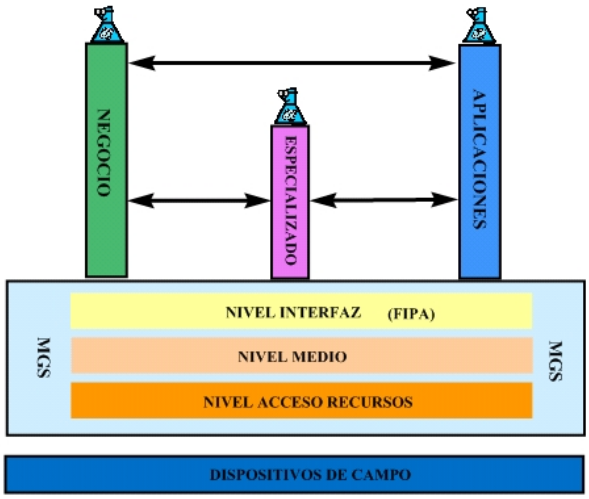


Figura 3. Automatización basada en Agentes.

1. El sistema operativo base que se utiliza, el cual puede ser cualquiera basado en el manejo de tiempo real. Hemos usado Linux más las extensiones tiempo real ofreecidas por RTAI [3] .
2. El nivel base, el cual contiene dos subsistemas: el que maneja todas las comunicaciones entre los diferentes sitios (Communication Manager), y el que permite administrar los agentes en la plataforma computacional (Agent Manager), el cual, entre otras funcionalidades, permite localizar agentes y asociar agentes a procesos. Este nivel es el que le confiere las características distribuidas al sistema e implementa todas las propiedades que se definen para el MGS, para los niveles de acceso a recurso y nivel medio, tales como migración, interoperatividad, nombramiento, etc. Es importante señalar que muchas de las funcionalidades y librerías del sistema operativo (LINUX) permiten implantar este nivel base.
3. El nivel interfaz, el cual esta compuesto por los agentes definidos por la especificación FIPA para brindarles servicios a las comunidades de agentes (SMA). Los agentes en esta capa de interfaz son los que permiten que se puedan implementar SMA sobre las plataformas computacionales actuales.
4. El nivel superior, en el cual existen dos comunidades de agentes. Una de ella es la comunidad de Agentes de Procesos, los cuales describen los diferentes objetos del medio de producción (abstracciones lógicas y funcionales de los procesos reales). La otra comunidad es la de los Agentes de Aplicaciones y Especializados, conformada por todos los agentes que realizan funciones específicas a nivel de supervisión, control, optimización, visualización, y otras aplicaciones especializadas y/o legadas. Esta comunidad brinda servicios a la comunidad de agentes de procesos. Ambas interactúan con el MGS a través de la capa de interfaz.

[3] Real Time Application Interface for Linux, *https://www.rtai.org/*

### *2.2 Control basado en Agentes*

Cabe destacar que las actividades de los agentes del nivel superior están basadas en los requerimientos de control de procesos. Estos, a su vez, establecen la necesidad de comunicaciones y tomas decisiones en tiempo real, adecuándose a situaciones no previstas mediante la detección de eventos, [Ríos-Bolívar et al., 2006]. En el diseño del nivel superior podrían estar contemplados los siguientes agentes:

1. Agente Proceso: Modelan los elementos de las unidades de producción. Cada unidad de producción está representada por un Agente Proceso. La composición de un Agente Proceso está basada, por un lado, en una división física del proceso, y por otro lado, en una división funcional de las tareas del agente. Así, un Agente Proceso podría representar desde dispositivos con capacidades de funcionamiento limitadas, como los sensores, actuadores u otros elementos de instrumentación de campo, hasta procesos complejos, tales como una unidad de producción petrolera, una caldera, etc.
2. Agente Control: Su tarea fundamental se inspira en la estabilidad y desempeño del proceso controlado. Realiza tareas de entonación, planificación y ejecución de las políticas de control. Su funcionamiento depende los siguientes agentes: Agente Diseñador del Control, Agente Ejecutor del Control y Agente Evaluador del Control.
3. Agente Diseñador del Control: Este agente se encarga de diseñar y/o ajustar planes de control a ejecutar sobre un horizonte de tiempo finito que garanticen el buen desempeño del proceso de producción, en términos de los requerimientos de control (estrategias específicas de control y parámetros de controlador) y de los requerimientos de procesamiento de control.
4. Agente Ejecutor del Control: Este agente genera las órdenes de control según los lineamientos estipulados en los planes actuales de control y desempeño.
5. Agente Evaluador del Control: Este agente se encarga de determinar el desempeño de los planes del control y controladores en ejecución, en términos del cumplimiento de los objetivos planteados en el diseño.

De esta manera, es posible conformar un ambiente de control de procesos basados en agentes. La figura 4 muestra la forma de estructura dicho ambiente.

A partir de las inter-relaciones entre los diferentes agentes, es fundamental la comunicación entre ellos bajo los

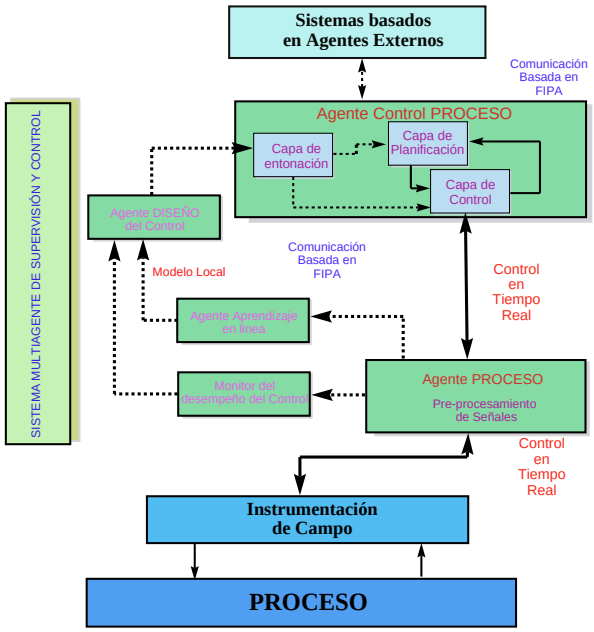


Figura 4. Implantación de control con agentes.

siguientes aspectos: la comunicación en tiempo real y la comunicación por eventos. Estos mecanismos de comunicación deben estar soportados por MGS.

## 3. MEDIO DE GESTIÓN DE SERVICIOS (MGS)

El Medio de Gestión de Servicios (MGS) es el conjunto básico de módulos de software que implantan las abstracciones mínimas para la especificación, implantación y manipulación de agentes y objetos. La especificación FIPA define la plataforma de agentes como un sistema constituido por los recursos de hardware y software (sistema operativo, software de comunicaciones, software de gestión de agentes) necesarios para que los agentes puedan ser desarrollados y usados.

### *3.1 Arquitectura de Implantación del MGS*

El MGS esta compuesto por 3 niveles. El Nivel Interfaz que define la interfaz entre el SMA y los componentes del sistema distribuido. Esta constituido por cinco agentes: Agente Administrador de Agentes (AAA), Agente Gestor de Recursos (AGR), Agente Gestor de Aplicaciones (AGA), Agente Gestor de Datos (AGD) y Agente de Control de Comunicación (ACC). Los agentes AGR, AGA y AGD son especializaciones del Directorio Facilitador (DF) especificado en FIPA, (Ver Figura 5). El nivel interfaz se encarga de establecer las pautas de conversación entre los componentes del sistema distribuido y el SMA. El Nivel Medio constituye el núcleo del sistema distribuido, provee servicios de software que requieren los agentes para poder interactuar entre sí y con el nodo de ejecución. Proporciona transparencia y seguridad en las transacciones, interoperabilidad de las aplicaciones y componentes de software, migración de agentes, objetos y/o recursos, comunicación interprocesos, localización de recursos (agentes y objetos) y provee un sistema de nombramiento para la localización de agentes y/o objetos. Finalmente, el Nivel de Acceso a Recursos esta integrado por el núcleo básico del Sistema Operativo, el cual maneja las funcionalidades de tiempo real y manejadores de acceso a hardware especifico que requiera el sistema. De esta forma, el MGS pretende conformidad arquitectural entre el estándar FIPA y la base de su implantación.

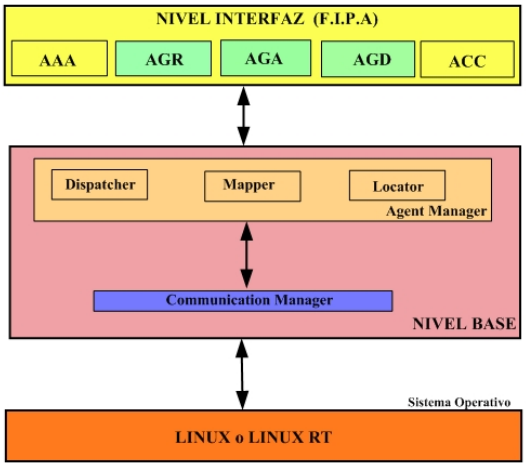


Figura 5. Medio de Gestión de Servicios.

### *3.2 Nivel Interfaz*

El nivel interfaz brinda todos los servicios requeridos por las comunidades de agentes para poder operar como SMA. Este nivel está conformado por los siguientes agentes:

**Administrador de Agentes (AAA).**
**Tipo**: Agente de Software.
**Papel**: Administrador del sistema multiagentes.
**Descripción**: se encarga de manejar, integrar y supervisar el estado del sistema multiagente. Este agente conoce la localización y estado de todos los agentes que existan en el sistema. El AAA dirige las migraciones de los agentes; así, cada agente que se mueve de un nodo a otro debe notificar al AAA el movimiento que ha efectuado; de manera que el agente administrador siempre tenga una vista ajustada al estado del sistema en tiempo real.

**Gestor de Datos (AGD).**
**Tipo**: Agente de Software.
**Papel**: Gestionar el manejo de Datos.
**Descripción**: este agente se encarga de establecer el enlace con los lugares donde existan datos de interés para el proceso que se esté ejecutando, sea que estos datos provengan de bases de datos (relacionales, orientadas a objetos, tiempo real, etc.), de SCADAS, DCS, medidores, o cualquier otro dispositivo o aplicación que pueda almacenar datos. Responde a las peticiones de los agentes del nivel superior.

**Gestor de Aplicaciones (AGA).**
**Tipo**: Agente de Software.
**Papel**: Localizador de aplicaciones.
**Descripción**: este agente se encarga de ubicar las aplicaciones que puedan ser requeridas por un proceso que se esté ejecutando, como por ejemplo de acceso a redes, programas de cálculo numérico o simbólico, aplicaciones de inteligencia artificial, de envío y recepción de mensajes, etc. Dichas aplicaciones pueden estar en cualquier servidor al que se tenga acceso y son requeridas por otros agentes de la comunidad.

**Gestor de Recursos (AGR)**.
**Tipo**: Agente de Software.
**Papel**: Gestionar los recursos del sistema.
**Descripción**: este agente se encarga de manejar, y llevar control del uso de los dispositivos necesarios en la ejecución de un proceso, como por ejemplo procesadores, dispositivos de entrada/salida, dispositivos de almacenamiento, etc.

**Control de Comunicación (ACC)**.
**Tipo**: Agente de Software.
**Papel**: Administrador de comunicaciones.
**Descripción**: es el encargado de mantener y controlar la comunicación entre sistemas multiagentes. Se encarga de mantener un estado confiable del canal de comunicación. Para llevar a cabo sus tareas usa directamente los servicios que provee el nivel base.

### *3.3 Nivel Base*

A partir de esas funcionalidades se define un sistema de base. Se asume Linux en sus versiones tradicionales y su versión tiempo real. Por esa razón, entre otras, e inicialmente por simplicidad, se asume que los agentes residen en procesos Linux. Cada proceso debe enlazar e invocar una biblioteca (*library*) que implanta las llamadas al MGS. Esto se realiza en cada proceso en lo que denominaremos *Ejecutivos*.

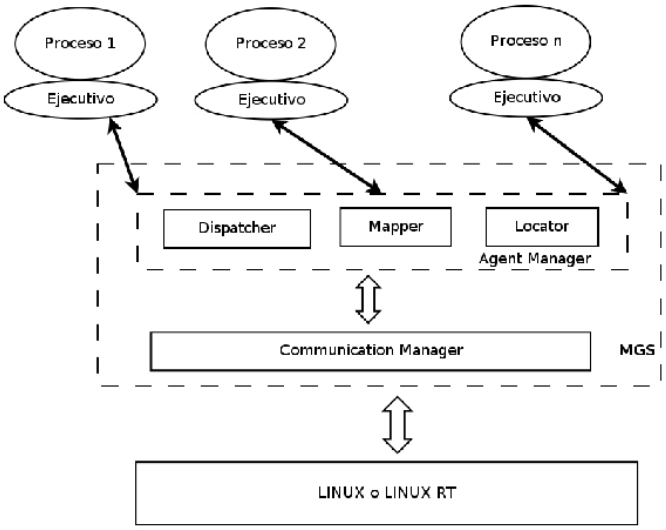


Figura 6. Arquitectura del núcleo operacional.

EL MGS en sus niveles inferiores está estructurado en dos módulos que deben ser instanciados, obligatoriamente, por los agentes del Nivel Interfaz (FIPA) para la realización de sus actividades: el manejador de agentes y el manejador de comunicación.

1. *Agent Manager* (Manejador de Agentes): Se encarga de corresponder agentes hacia procesos Linux. Contempla funciones como creación, destrucción y manejo de recursos del sistema operativo para la manipulación de agentes. La creación de identificadores únicos sería también su responsabilidad. El Agent Manager también debe implantar la invocación de agentes bajo los esquemas estáticos y dinámicos. Los esquemas de migración, intra-sitio o inter-sitio son gestionados por este módulo; de igual manera, la localización de agentes. Este módulo está estructurado en los tres sub-módulos siguientes:
   *a*) *Despachador*: Se encarga de despachar invocaciones a los agentes. Del lado superior, recibe invocaciones desde los procesos y las hace llegar al despachador remoto a través del manejador de comunicación. Del lado inferior, recibe invocaciones remotas y se las hace llegar al proceso que contiene el agente. El despachador puede implantar despacho dinámico; es decir, un agente puede construir, en tiempo de ejecución, infraestructura para recibir invocaciones.
   *b*) *Mapper*: se encarga de otorgar identificadores únicos y de gestionar los recursos del sitio para los agentes y procesos. Este módulo es pues responsable de la creación y destrucción de agentes. Similarmente, este módulo gestiona la migración de agentes.
   *c*) *Localizador*: Se encarga de localizar agentes respecto a sus identificadores únicos.
2. Communication Manager (Manejador de Comunicación): Este módulo se encarga de proveer comunicación confiable de red orientada a invocación. La semántica queda a decidir entre "*a lo más una vez*.º "*exactamente una vez*", según las suposiciones de fallas que se consideren para los agentes. Probablemente, los componentes del MGS estarán implantados mediante procesos privilegiados Linux.

### *3.4 Definiciones y especificaciones generales*

Para comprender la propuesta de implantación del MGS y su relación con la especificación FIPA se utilizan ciertas definiciones que caracterizan la estructura de implementación, tales como Objeto, Interfaces, Identificadores únicos, etc.

1. Objeto: Un objeto es una especificación encapsulada de un ente, bajo un tipo de dato abstracto, donde se oculta completamente su implantación y se define claramente su interfaz de operación.
2. Interfaces: Los agentes y objetos poseen una interfaz definida por una clase de objeto compatible con la especificación CORBA IDL o CORBA UML, para garantizar la integración de los mismos.
3. Identificadores únicos: Un identificador único es un objeto de identificación de instancias de objetos y de agentes cuya identidad es única en el espacio y en el tiempo. Todo objeto o agente del sistema posee un identificador único. Un identificador único puede convertirse a una cadena de caracteres ASCII culminada en el valor de byte "0". El MGS es responsable de generar instancias **Uid**.
4. Fin: Un fin es una abstracción que representa un flujo de ejecución. Tal flujo se destina a implantar el fin (goal) para el cual el agente haya sido programado.

*Agentes* Un agente es un ente obrante cuyas funciones vislumbrables son:

1. Implantar un objeto.
2. Especificar uno o más fines.

En general, se puede decir que los objetos son objetivos en el sentido de que sólo se considera su interfaz, su cara exterior. Por otro lado, en los agentes, se considera tanto lo interior como lo exterior.

La clase abstracta $Abstract_Agent$ define al agente fundamental del cual derivan directamente dos especializaciones de agentes: agente global y agente proceso.

Un agente proceso es aquel cuyo acceso está restringido sólo al proceso que lo instancia. Este agente puede acceder a cualquier agente global del sistema, pero ningún agente fuera del proceso puede accederlo. Un agente global es aquél que tiene ámbito global (distribuido). Es decir, es referenciable por cualquier otro ente del sistema.

Los agentes globales se clasifican en dos tipos: proactivo y reactivo. En términos sistema, un agente reactivo requiere ser activado externamente para cumplir un fin, mientras que uno proactivo sea activa autónomamente para cumplir uno o más fines. En un agente global, el mecanismo de acceso es controlado a través de su **UID**.

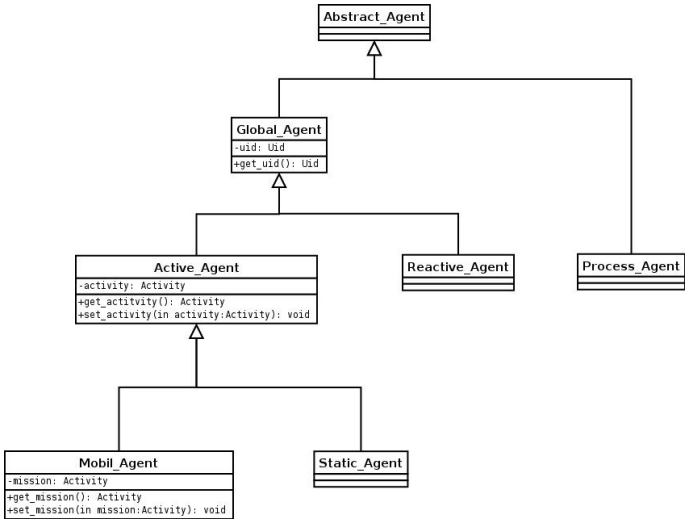


Figura 7. Jerarquía de Clases en el Sistema de Agentes.

Pueden existir agentes que no tengan "fin", en este caso, el fin será identificado como $Null\_End$. Un agente cuyo fin no sea $Null\_End$ puede contener más de un fin. Los fines pueden obtenerse mediante el método *get_end*, el cual retorna el fin según el orden de inserción.

*Agentes móviles* Un agente móvil es aquél que puede desplazarse entre los procesos de un sistema. La decisión de migrar la debe tomar el propio agente mediante el método *migrate*. Los agentes móviles deben tener una "misión"; esto es, un fin a realizarse a su llegada a destino bajo su responsabilidad. Así, la misión es obligatoria y debe especificarse en tiempo de instanciación, pero puede modificarse, una sola vez, luego de una migración, mediante el método *set_mission*.

*Comunicación* El MGS maneja la comunicación de mensajes fiable entre agentes. Por fiable se entiende un mensaje que llega a su destino, íntegro, que no se duplica, o una excepción de notificación de falla. En lo que sigue, el agente que origina la comunicación se denomina remitente, mientras que el que la destina (la recibe) destinatario.

El MGS maneja los esquemas de comunicación siguientes: síncrona o "$Rendez-Vous$", asíncrona, y semántica "RPC". La comunicación "$Rendez-Vous$"se utiliza para situaciones en las que haya que garantizar correspondencia de orden entre emisión y recepción. Bajo los otros esquemas, el orden de recepción no necesariamente corresponde con el de emisión.

Un mensaje tiene dos facetas. La primera, representada por la clase $Message$, concierne al mensaje dentro de la actividad del agente interesado. En este contexto, hay cuatro especializaciones de $Message$:

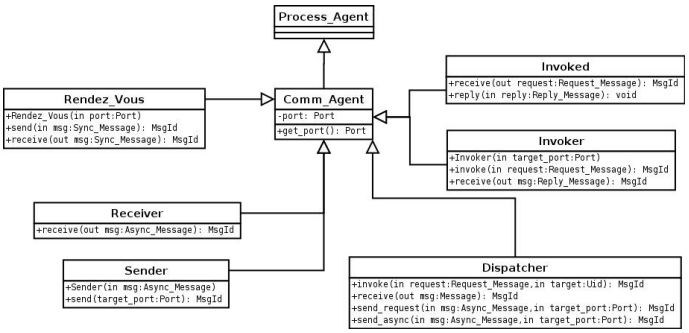


Figura 8. Características de los Mensajes.

1. $Sync\_Message$: Modela un mensaje requerimiento correspondiente a una comunicación $Rendez-Vous$ en la que los flujos del remitente y destinatario se bloquean hasta que no se consuma el mensaje.
2. $Async\_Message$: Modela un mensaje asíncrono; el remitente entrega el mensaje al sistema de comunicación y se desbloquea.
3. $Request\_Message$: Concierne a un mensaje de requerimiento (*request*) correspondiente a una comunicación RPC en la que el remitente se bloquea hasta recibir respuesta explícita del destinatario.
4. $Reply\_Message$: Concierne a un mensaje de respuesta del destinatario a un mensaje previo de requerimiento, para una comunicación RPC.

Una vez que el mensaje es entregado al sistema de comunicación, aparece la segunda faceta de manipulación de mensajes representada bajo la clase *MsgId*. Esta clase modela aspectos de un mensaje que se conocen después de que el sistema de comunicación haya sido informado del mensaje. Lo más representativo e importante de esta clase (*MsgId*) es que ella contiene un identificador único del mensaje. Este identificador ayuda a convalidar el mensaje e identificarlo unívocamente en las situaciones en las que puedan solaparse varios mensajes.

Para manejar la comunicación se debe instanciar algún agente de comunicación de entre aquellos que se muestran la Figura 9.

1. Comunicación $Rendez-Vous$ (Síncrona) Los mensajes síncronos se manipulan mediante dos instancias del agente $Rendez-Vous$. El constructor de esta clase acepta el **UID** del otro extremo (remitente o destinatario).
2. Comunicación Asíncrona Los mensajes asíncronos usan dos agentes diferentes. El remitente debe instanciar un agente Sender mientras que el destinatario debe instanciar uno Receiver.

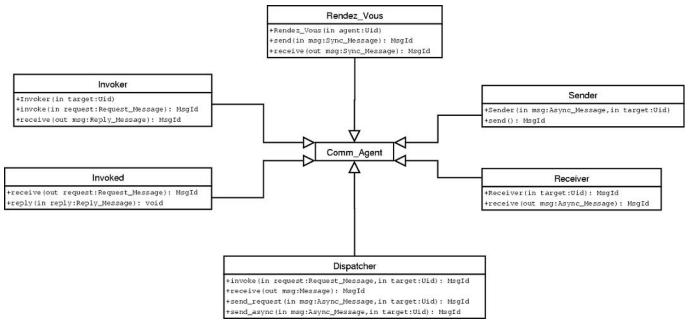


Figura 9. Agente de Comunicación.

3. Comunicación RPC La comunicación tipo RPC se modela mediante los agentes *Invoker* e *Invoked*. El destinatario debe colocarse a la escucha con un agente Invoked y llamar al método receive, el cual se bloquea hasta recibir un requerimiento. El remitente instancia un agente *Invoker* y llama al método *invoke* para enviar el requerimiento.

   Cuando un agente *Invoked* recibe un requerimiento, éste se desbloquea. Cuando se considere oportuno, el agente *Invoked* responderá mediante el método *reply* indicando el *MsgId* del mensaje al que se está respondiendo.

   Puesto que el orden de recepción no necesariamente corresponde con el de emisión, la recepción del *reply* está separada bajo el método *receive*. El *reply* toma el *MsgId* del request. De esta manera, el remitente puede identificar el requerimiento del cual está recibiendo respuesta.

## 4. CONCLUSIONES

Considerando la filosofía de agentes inteligentes, hemos presentado una plataforma para la implantación de aplicaciones de control y supervisión de procesos basada en agentes inteligentes. A partir de una arquitectura de automatización integrada y un marco de desarrollo de la plataforma de agentes, las aplicaciones de control se definen a través de procesos de coordinación, ejecución y evaluación de las tareas de supervisión y control, necesarias para el procesamiento de la información del proceso y para la toma de decisiones. El control basado en agentes inteligentes considera la captura de la información de los procesos a través de un modelo operacional de los mismos, caracterizado por un Agente Proceso.

Para el procesamiento de la información es necesario las comunicaciones entre los diferentes actores, para lo cual se ha diseñado un Medio de Gestión de Servicios (MGS), que permite soportar actividades de tiempo real y el manejo de operaciones basadas en eventos.

## REFERENCIAS

J. Aguilar, A. Rios, F. Rivas, O. Teran, L. Leon, and N. Perez. Definición de dominios y paradigmas en una arquitectura de automatización industrial. Technical Report 05-04, Fundacite-Mérida, Mérida, 2004.

J. Aguilar, M. Cerrada, and F. Hidrobo. A methodology to specify multiagent systems. *Lecture Notes in Artificial Intelligence, Springer-Verlag*, 4496:92–101, 2007.

Martin Albert, Thomas Längle, Heinz Wörn, Michele Capobianco, and Attilio Brighenti. Multi-agent systems for industrial diagnostics. In *XV IFAC World Congress*, Barcelona, Spain, 2003.

F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. JADE: A white paper. Technical Report 3, TILAB, http://exp.telecomitalialab.com, 2003.

Alexei Bratoukhine, Yoseba Penya, and Thilo Sauter. Intelligent software agents in plant automation. Technical report, Vienna University Of Technology, Wien Austria, 2002.

C. Bravo, J. Aguilar, F. Rivas, and M. Cerrada. Design of an architecture for industrial automation based on multi-agents systems. In *Proceeding of the 16th IFAC World Congress*, Prague, 2005.

F. Hidrobo, A. Ríos-Bolívar, J. Aguilar, and L. León. An architecture for industrial automation based on intelligent agents. *WSEAS Transaction on Computers*, 4(12):1808–1815, 2005.

Nicholas Jennings and Stefan Bussmann. Agent-based control systems. *IEEE Control Systems Magazine*, June: 61–73, 2003.

OMG. Unified Modeling Language specification, 2003. URL `http://www.omg.org`. Version 2.0, June 2003 via `http://www.omg.org`.

Antti Pakonen. Information agent technology in process automation systems. Master's thesis, Helsinki University of Technology, Espoo Finland, 2004.

Jim Pinto. Instrumentation & control on the frontiers of a new millennium. *Journal Instruments & Control Systems*, 1(2):78–90, 2000.

A. Ríos-Bolívar, F. Hidrobo, M. Cerrada, and J. Aguilar. Control and supervision system development with intelligent agents. *WSEAS Transactions on Systems*, 6(1): 141–148, 2007.

A. Ríos-Bolívar, M. Cerrada, F. Hidrobo, J. Aguilar, and J. Durán. Towards control and supervision systems based on intelligent agents. In *Proc. of 2nd WSEAS International Conference on Dynamical Systems and Control*, pages 36–41, Bucharest, Romania, 2006.

Ilkka Seilonen, Teppo Pirttioja, and Pekka Appelqvist. Agent technology and process automation. Technical report, Helsinki University of Technology, 2003.

Thomas Wagner. Applying agents for engineering of industrial automation systems. In M. Schillo *et al., editor,* MATES 2003*, page 62–73. Springer-Verlag, Berlin Heidelberg, 2003.*

*G. Weiss.* Multiagent Systems*. The MIT Press, 1999.*